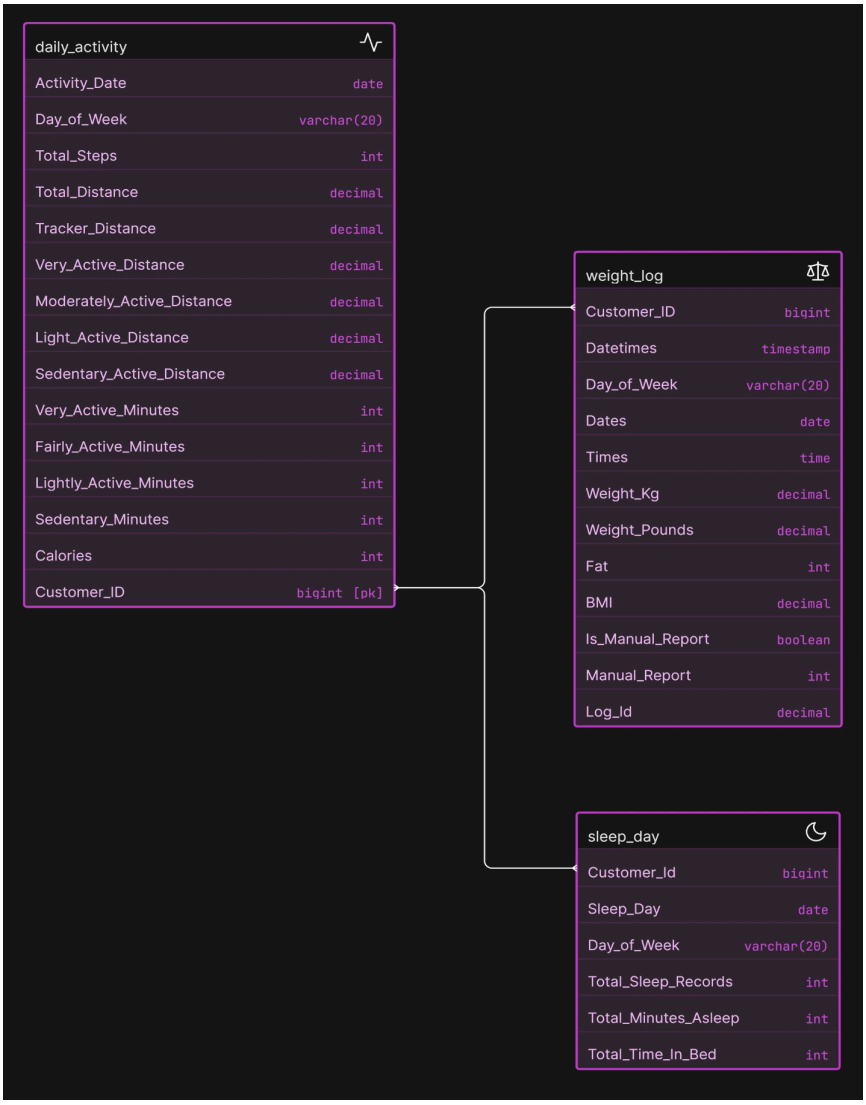
BellaBeat Fitness Tracker

Activity, Sleep, and Weight Logs: SQL Analysis

This report presents SQL queries and their results derived from the fitness log tables: daily_activity, sleep_day, and weight_log.

Data Check



Initial count of records per table:

Table	Count
daily_activity	940
sleep_day	410
weight_log	60

1. Active Days Analysis

Problem: Identify the day of the week when the customers are **most active** and **least active**, based on the total number of steps.

Solution:

```
WITH active_days AS(
SELECT
    day_of_week
    , SUM(total_steps) AS total_steps_sum
    , FIRST_VALUE(day_of_week) OVER (ORDER BY SUM(total_steps) DESC) AS
most_active
    , FIRST_VALUE(day_of_week) OVER (ORDER BY SUM(total_steps)) AS least_active
FROM daily_activity
GROUP BY day_of_week
)
SELECT
    DISTINCT most_active
    , least_active
FROM active_days;
```

Output:

```
| most_active | least_active |
| ----------- / ------------ /
| Tuesday     | Sunday       |
```

2. Most Effective Sleeper

Problem: Identify the customer who has the **most effective sleep**. Effective sleep is determined by the customer who spends the least amount of wasted time (time in bed without sleeping).

Solution:

```
WITH effectiveness AS(
SELECT
        customer_id
        , (SUM(total_time_in_bed) - SUM(total_minutes_asleep)) AS wasted_time
        , RANK() OVER(ORDER BY
                (SUM(total_time_in_bed) - SUM(total_minutes_asleep))) AS effctv_rank
FROM sleep_day
GROUP BY customer_id
)
SELECT
        customer_id
FROM effectiveness
WHERE effctv_rank = 1;
```

Output:

```
| customer_id |
| ----------- |
| 7007744171  |
```

3. Customers with No Sleep Record

Problem: Identify customers who are present in the daily_activity log but have **no corresponding sleep record** in the sleep_day table.

Solution:

```sql
-- Correlated Subquery
SELECT
      DISTINCT d.customer_id
FROM daily_activity d
WHERE NOT EXISTS (
      SELECT
            customer_id
      FROM sleep_day s
      WHERE s.customer_id = d.customer_id
);

-- NOT IN Method
SELECT DISTINCT CUSTOMER_ID
FROM DAILY_ACTIVITY
WHERE CUSTOMER_ID NOT IN (
    SELECT CUSTOMER_ID
    FROM SLEEP_DAY
);

-- We can also solve it using JOIN and NULL
```

Output:

```
| customer_id |
| ----------- |
| 1624580081  |
| 4057192912  |
| 8253242879  |
| 3372868164  |
| 8877689391  |
| 6290855005  |
| 2873212765  |
| 2022484408  |
| 8583815059  |
```

4. Customers with All Three Logs

Problem: Fetch all customers whose daily activity, sleep, and weight logs are all present.

Solution:

```
SELECT CUSTOMER_ID FROM daily_activity
INTERSECT
SELECT CUSTOMER_ID FROM weight_log
INTERSECT
SELECT CUSTOMER_ID FROM sleep_day;

-- OR

SELECT DISTINCT da.customer_id
FROM daily_activity da
JOIN weight_log wl
   ON da.customer_id = wl.customer_id
JOIN sleep_day sd
   ON da.customer_id = sd.customer_id;
```

Output:

```
| customer_id |
| ----------- |
| 4558609924  |
| 6962181067  |
| 1503960366  |
| 4319703577  |
| 5577150313  |
| 1927972279  |
```

5. Total Sleep PIVOTED by Day of Week

Problem: For each customer, display the total minutes they slept for each day of the week. The output should contain 8 columns: customer_id and the 7 days of the week.

Solution:

```sql
-- solution using CASE statement
SELECT customer_id,
       SUM(CASE WHEN day_of_week = 'Monday'    THEN total_minutes_asleep ELSE 0
END) AS monday,
       SUM(CASE WHEN day_of_week = 'Tuesday'   THEN total_minutes_asleep ELSE 0
END) AS tuesday,
       SUM(CASE WHEN day_of_week = 'Wednesday' THEN total_minutes_asleep ELSE 0
END) AS wednesday,
       SUM(CASE WHEN day_of_week = 'Thursday'  THEN total_minutes_asleep ELSE 0
END) AS thursday,
       SUM(CASE WHEN day_of_week = 'Friday'    THEN total_minutes_asleep ELSE 0
END) AS friday,
       SUM(CASE WHEN day_of_week = 'Saturday'  THEN total_minutes_asleep ELSE 0
END) AS saturday,
       SUM(CASE WHEN day_of_week = 'Sunday'    THEN total_minutes_asleep ELSE 0
END) AS sunday
FROM sleep_day
GROUP BY customer_id
ORDER BY customer_id;

-- solution using CROSSTAB
CREATE EXTENSION tablefunc;

SELECT customer_id,
       COALESCE(monday, 0)    AS monday,
       COALESCE(tuesday, 0)   AS tuesday,
       COALESCE(wednesday, 0) AS wednesday,
       COALESCE(thursday, 0)  AS thursday,
       COALESCE(friday, 0)    AS friday,
       COALESCE(saturday, 0)  AS saturday,
       COALESCE(sunday, 0)    AS sunday
FROM CROSSTAB(
        'SELECT customer_id, day_of_week, SUM(total_minutes_asleep) AS
total_sleep
         FROM sleep_day
         GROUP BY customer_id, day_of_week
```

```
         ORDER BY customer_id, day_of_week',
        'SELECT DISTINCT day_of_week FROM sleep_day'
    )
    AS result(
        customer_id bigint,
        monday      bigint,
        tuesday     bigint,
        wednesday   bigint,
        thursday    bigint,
        friday      bigint,
        saturday    bigint,
        sunday      bigint
    );

-- Solution for SQL Server (T-SQL)
SELECT customer_id,
       COALESCE([Monday], 0)    AS monday,
       COALESCE([Tuesday], 0)   AS tuesday,
       COALESCE([Wednesday], 0) AS wednesday,
       COALESCE([Thursday], 0)  AS thursday,
       COALESCE([Friday], 0)    AS friday,
       COALESCE([Saturday], 0)  AS saturday,
       COALESCE([Sunday], 0)    AS sunday
FROM (
    SELECT customer_id, day_of_week, total_minutes_asleep
    FROM sleep_day
) src
PIVOT (
    SUM(total_minutes_asleep)
    FOR day_of_week IN ([Monday], [Tuesday], [Wednesday], [Thursday], [Friday],
[Saturday], [Sunday])
) p
ORDER BY customer_id;
```

Output:

customer_id	monday	tuesday	wednesday	thursday	friday	saturday	sunday
1503960366	1532	1436	938	1087	1029	2093	892
1644430081	0	124	0	119	0	137	796
1844505072	0	722	0	644	0	590	0
1927972279	1046	0	166	475	398	0	0
2026352035	1358	2130	2459	2057	2561	2060	1548
2320127002	0	61	0	0	0	0	0
2347167796	901	374	1313	1268	915	998	933

3977333714	1347	992	1202	1435	1009	1333	904
4020332650	1265	77	226	385	478	364	0
4319703577	1927	2195	1393	1971	1338	1553	2016
4388161847	1002	1785	1341	581	1719	2247	529
4445114986	1992	1185	2241	1205	1977	898	1287
4558609924	103	0	126	171	0	238	0
4702921684	2046	1846	2069	1090	2052	1071	1098
5553957443	2008	2499	2034	1549	2370	2093	1815
5577150313	2234	1202	1326	1189	2337	1471	1473
6117666160	465	1370	1293	1738	1016	1751	985
6775888955	0	0	423	391	235	0	0
6962181067	2146	1605	2382	1671	2397	1890	1797
7007744171	0	79	0	0	0	58	0
7086361926	1927	1015	1660	1127	2283	1550	1313
8053475328	0	405	0	0	486	0	0
8378563200	2022	2103	2203	1804	2168	2003	1496
8792009665	974	682	888	1152	1921	503	415

6. Highest and Lowest Weight Dates

Problem: For each customer, display the date when they had the highest weight and the date when they had the lowest weight (including the weight in kilograms).

Solution:

```
SELECT DISTINCT d.customer_id,
COALESCE(FIRST_VALUE(dates || '  (' || weight_kg || ' kgs)')
              OVER (PARTITION BY d.customer_id ORDER BY weight_kg DESC), 'NA') AS
highest_weight_on,
      COALESCE(FIRST_VALUE(dates || '  (' || weight_kg || ' kgs)')
              OVER (PARTITION BY d.customer_id ORDER BY weight_kg), 'NA') AS
lowest_weight_on
FROM weight_log w
RIGHT JOIN daily_activity d ON d.customer_id = w.customer_id
ORDER BY highest_weight_on;
```

Output:

```
| customer_id | highest_weight_on      | lowest_weight_on       |
| ----------- | ---------------------- | ---------------------- |
| 6962181067  | 2016-04-12 (62.5 kgs)  | 2016-05-03 (61 kgs)    |
| 1927972279  | 2016-04-13 (133.5 kgs) | 2016-04-13 (133.5 kgs) |
| 4319703577  | 2016-04-17 (72.4 kgs)  | 2016-05-04 (72.3 kgs)  |
| 5577150313  | 2016-04-17 (90.7 kgs)  | 2016-04-17 (90.7 kgs)  |
| 8877689391  | 2016-04-18 (85.8 kgs)  | 2016-05-12 (84 kgs)    |
| 4558609924  | 2016-04-25 (70.3 kgs)  | 2016-05-09 (69.1 kgs)  |
| 1503960366  | 2016-05-02 (52.6 kgs)  | 2016-05-02 (52.6 kgs)  |
| 2873212765  | 2016-05-12 (57.3 kgs)  | 2016-04-21 (56.7 kgs)  |
| 2026352035  | NA                     | NA                     |
| 2320127002  | NA                     | NA                     |
| ...........| ...........           | ...........            |
```

7. Day with Most Sleep

Problem: Fetch the day when customers collectively slept the most (highest total minutes asleep).

Solution:

```
WITH most_sleep AS(
SELECT
      day_of_week
      , SUM(total_minutes_asleep) AS total_sleep_time
      , RANK() OVER(ORDER BY SUM(total_minutes_asleep) DESC) AS rnk
FROM sleep_day
GROUP BY day_of_week
)
SELECT
      day_of_week
FROM most_sleep
WHERE rnk = 1;
```

Output:

```
| day_of_week |
| ----------- |
| Wednesday   |
```

8. Percentage of Time Spent Sleeping vs. Lying in Bed

Problem: For each day of the week, determine the percentage of time customers spent sleeping relative to their total time in bed.

Solution:

```
SELECT
      day_of_week
      ,(CAST(SUM(total_time_in_bed) AS DECIMAL) - CAST(SUM(total_minutes_asleep)
AS DECIMAL)) AS time_in_bed_without_sleep
      ,ROUND((CAST(SUM(total_minutes_asleep) AS DECIMAL) /
CAST(SUM(total_time_in_bed) AS DECIMAL)) * 100,2) AS pct_time_asleep
FROM sleep_day
GROUP BY day_of_week
ORDER BY 3 DESC;
```

Output:

day_of_week	time_in_bed_without_sleep	pct_time_asleep
Wednesday	2333	92.48
Thursday	2149	92.28
Monday	1741	91.72
Tuesday	2519	91.26
Saturday	2324	91.13
Friday	2259	91.10
Sunday	2792	89.92

9. Most Frequently Logged Day of Week

Problem: Identify the most repeated day of the week across all log entries (activity, sleep, and weight).

Solution:

```sql
WITH mention_of_day_from_all_tables AS (
SELECT day_of_week FROM daily_activity -- 940
UNION ALL
SELECT day_of_week FROM sleep_day -- 410
UNION ALL
SELECT day_of_week FROM weight_log -- 60
)
SELECT
      day_of_week
      , COUNT(1) AS repetition_count
FROM mention_of_day_from_all_tables
GROUP BY day_of_week
ORDER BY repetition_count DESC;

-- OR Getting to the actual answer with just top day
WITH all_days AS (
    -- Combine all day_of_week entries from all tables
    SELECT day_of_week FROM daily_activity
    UNION ALL
    SELECT day_of_week FROM weight_log
    UNION ALL
    SELECT day_of_week FROM sleep_day
),
day_counts AS (
    -- Count occurrences per day and rank them
    SELECT day_of_week,
           COUNT(1) AS occurrence,
           RANK() OVER (ORDER BY COUNT(1) DESC) AS rank_by_frequency
    FROM all_days
    GROUP BY day_of_week
)
SELECT day_of_week AS most_repeated_day_of_week
FROM day_counts
WHERE rank_by_frequency = 1;
```

Output:

```
| most_repeated_day_of_week |
| ------------------------- |
| Wednesday                 |
```

10. Average Distance for Active Customers

Problem: Based on the given data, for each customer, identify the average distance (in kilometers) they walked on days when they took more than 6,000 steps.

Solution:

```sql
SELECT customer_id,
       ROUND(AVG(total_distance), 2) AS distance_kms
FROM daily_activity
WHERE total_steps > 6000
GROUP BY customer_id
ORDER BY distance_kms DESC;
```

Output:

```
| customer_id | distance_kms |
| ----------- | ------------ |
| 8877689391  | 13.53        |
| 8053475328  | 12.48        |
| 7007744171  | 9.78         |
| 4388161847  | 8.76         |
| 2022484408  | 8.28         |
| 8378563200  | 8.22         |
| 5553957443  | 8.11         |
| 1503960366  | 8.07         |
| 6962181067  | 7.81         |
| 3977333714  | 7.76         |
| 6117666160  | 7.71         |
| 4702921684  | 7.57         |
| 8583815059  | 7.54         |
| 5577150313  | 7.52         |
| 7086361926  | 7.51         |
| 2347167796  | 7.37         |
| 1644430081  | 7.31         |
| 1624580081  | 6.75         |
| 4020332650  | 6.58         |
```

8253242879	6.41
4319703577	6.41
6775888955	5.90
4558609924	5.76
3372868164	5.49
6290855005	5.49
2873212765	5.38
2026352035	5.14
2320127002	4.93
1844505072	4.89
4445114986	4.82
8792009665	4.63