

STRAVA FITNESS DATA ANALYTICS CASE-STUDY

SQLite analytics

dailyActivity_merge.csv

Use the dailyActivity_merge.csv

```
9  -- STEP 1: Cleaning & Preprocessing in SQLite
10  / 1.1. Check for NULL or Missing Values
11  SELECT
12    SUM(CASE WHEN Id IS NULL THEN 1 ELSE 0 END) AS NullId,
13    SUM(CASE WHEN ActivityDate IS NULL THEN 1 ELSE 0 END) AS NullActivityDate,
14    SUM(CASE WHEN TotalSteps IS NULL THEN 1 ELSE 0 END) AS NullTotalSteps,
15    SUM(CASE WHEN Calories IS NULL THEN 1 ELSE 0 END) AS NullCalories
16  FROM dailyActivity_merged;
17
```

NullId	NullActivityDate	NullTotalSteps	NullCalories
1	0	0	0

```
Execution finished without errors.
Result: 1 rows returned in 14ms
At line 3:
-- STEP 1: Cleaning & Preprocessing in SQLite
-- / 1.1. Check for NULL or Missing Values
SELECT
  SUM(CASE WHEN Id IS NULL THEN 1 ELSE 0 END) AS NullId,
  SUM(CASE WHEN ActivityDate IS NULL THEN 1 ELSE 0 END) AS NullActivityDate,
  SUM(CASE WHEN TotalSteps IS NULL THEN 1 ELSE 0 END) AS NullTotalSteps,
  SUM(CASE WHEN Calories IS NULL THEN 1 ELSE 0 END) AS NullCalories
FROM dailyActivity_merged;
```

```
21  -- 1.2. Check for Duplicates
22  SELECT Id, ActivityDate, COUNT(*) AS cnt
23  FROM dailyActivity_merged
24  GROUP BY Id, ActivityDate
25  HAVING cnt > 1;
26
```

Id	ActivityDate	cnt

```
Execution finished without errors.
Result: 0 rows returned in 10ms
At line 21:
-- 1.2. Check for Duplicates
SELECT Id, ActivityDate, COUNT(*) AS cnt
FROM dailyActivity_merged
GROUP BY Id, ActivityDate
HAVING cnt > 1;
```

```
28  -- 1.3. Check for Outliers or Invalids
29  /-- Negative steps, distance, or calories
30  SELECT *
31  FROM dailyActivity_merged
32  WHERE TotalSteps < 0 OR TotalDistance < 0 OR Calories < 0;
33
```

Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance	LightActiveDistance	SedentaryActiveDistance	VeryActiveMinutes	FairlyActiveMinutes

```
Execution finished without errors.
Result: 0 rows returned in 10ms
At line 28:
-- 1.3. Check for Outliers or Invalids
-- /-- Negative steps, distance, or calories
SELECT *
FROM dailyActivity_merged
WHERE TotalSteps < 0 OR TotalDistance < 0 OR Calories < 0;
```

```

36 -- 1.4. Convert ActivityDate to usable format (if not already)
37   -- Create a new column with ISO format
38   ALTER TABLE dailyActivity_merged ADD COLUMN ActivityDateFormatted TEXT;
39
40 UPDATE dailyActivity_merged
41 SET ActivityDateFormatted =
42   substr(ActivityDate, 7, 4) || '-' ||
43   substr(ActivityDate, 1, 2) || '-' ||
44   substr(ActivityDate, 4, 2);
45

```

Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance	LightActiveDistance	SedentaryActiveDistance	VeryActiveMinutes	FairlyActiveMinutes
<hr/>											
Execution finished without errors.											
Result: query executed successfully. Took 5ms, 940 rows affected											
At line 40:											
UPDATE dailyActivity_merged											
SET ActivityDateFormatted =											
substr(ActivityDate, 7, 4) '-'											
substr(ActivityDate, 1, 2) '-'											
substr(ActivityDate, 4, 2);											

Step 2: Insights and analytics Queries

```

49 -- STEP 2: Insights and Analytics Queries
50   -- 2.1. Average Steps, Calories, Distance
51
52   SELECT
53     ROUND(AVG(TotalSteps), 2) AS AvgSteps,
54     ROUND(AVG(TotalDistance), 2) AS AvgDistance,
55     ROUND(AVG(Calories), 2) AS AvgCalories
56   FROM dailyActivity_merged;
57

```

AvgSteps	AvgDistance	AvgCalories
1 7637.91	5.49	2303.61

Execution finished without errors.

--🧠 Insight: On average, users take around AvgSteps steps, cover AvgDistance km, and burn approximately AvgCalories calories daily.

```

60   -- 2.2. Users With Highest Average Activity
61   SELECT Id,
62     ROUND(AVG(TotalSteps), 2) AS AvgSteps,
63     ROUND(AVG(Calories), 2) AS AvgCalories
64   FROM dailyActivity_merged
65   GROUP BY Id
66   ORDER BY AvgSteps DESC
67   LIMIT 5;

```

Id	AvgSteps	AvgCalories
1 8877689391	16040.03	3420.26
2 8053475328	14763.29	2945.81
3 1503960366	12116.74	1816.42
4 2022484408	11370.65	2509.97
5 7007744171	11323.42	2544.0

Execution finished without errors.

--🧠 Insight: The top 5 most active users averaged the highest number of steps and calorie burn per day, indicating consistent high activity.

```
72 -- 2.3. Relationship Between Steps and Calories
73 SELECT TotalSteps, Calories
74 FROM dailyActivity_merged
75 ORDER BY TotalSteps DESC
76 LIMIT 20;
77
```

	TotalSteps	Calories
1	36019	2690
2	29326	4547
3	27745	4398
4	23629	3808
5	23186	3921
6	22988	3577
7	22770	4022
8	22359	3554
9	22244	2670
10	22026	3589
11	21727	4015
12	21420	3832
13	21129	3793
14	20669	3410
15	20500	3403

--🧠 Insight: As step count increases, calories burned also tend to increase, showing a positive correlation. This can be better visualized through a scatter plot.

```
79 --2.4. Active vs Sedentary Minutes Summary
80 SELECT
81   ROUND(AVG(VeryActiveMinutes), 2) AS AvgVeryActive,
82   ROUND(AVG(LightlyActiveMinutes), 2) AS AvgLightlyActive,
83   ROUND(AVG(SedentaryMinutes), 2) AS AvgSedentary
84 FROM dailyActivity_merged;
```

AvgVeryActive	AvgLightlyActive	AvgSedentary
21.16	192.81	991.21

Execution finished without errors.

--🧠 Insight: On average, users spend AvgVeryActive minutes being very active, AvgLightlyActive minutes lightly active, and AvgSedentary minutes being sedentary, showing a need for more active movement.

```
88 -- 2.5. Days with Over 10K Steps
89 SELECT COUNT(*) AS DaysOver10KSteps
90 FROM dailyActivity_merged
91 WHERE TotalSteps >= 10000;
92
```

DaysOver10KSteps
303

Execution finished without errors.

--🧠 Insight: The number of days users walked more than 10,000 steps shows how often they meet or exceed the daily recommended activity level.

```

95  -- 2.6. Analyzing Daily Activity Distribution Across Users
96  WITH user_avg_steps AS (
97      SELECT Id, AVG(TotalSteps) AS avg_steps
98      FROM dailyActivity_merged
99      GROUP BY Id
100 )
101  SELECT
102      CASE
103          WHEN avg_steps < 5000 THEN 'Sedentary (<5k)'
104          WHEN avg_steps BETWEEN 5000 AND 9999 THEN 'Moderately Active (5k-10k)'
105          ELSE 'Highly Active (10k+)'
106      END AS ActivityLevel,
107      COUNT(*) AS UserCount
108  FROM user_avg_steps
109  GROUP BY ActivityLevel;
110

```

ActivityLevel	UserCount
1 Highly Active (10k+)	7
2 Moderately Active (5k-10k)	18
3 Sedentary (<5k)	8

-- 🧠 Insight: This breakdown reveals the activity levels of users, highlighting how many are sedentary (<5k steps), moderately active (5k–10k), and highly active (>10k).

-- This can help in segmenting users for targeted fitness recommendations or challenges.

```

115  -- 2.7. Calorie Burn Efficiency: Calories per Distance vs Active Minutes
116  SELECT
117      Id,
118      ROUND(SUM(Calories)/SUM(TotalDistance), 2) AS CaloriesPerKM,
119      ROUND(SUM(Calories)/NULLIF(SUM(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes), 0), 2) AS CaloriesPerActiveMin
120  FROM dailyActivity_merged
121  GROUP BY Id
122  HAVING CaloriesPerKM IS NOT NULL AND CaloriesPerActiveMinute IS NOT NULL
123  ORDER BY CaloriesPerKM DESC
124  LIMIT 10;
125

```

	Id	CaloriesPerKM	CaloriesPerActiveMinute
1	1927972279	3424.35	53.0
2	8792009665	1653.79	20.0
3	4020332650	1467.17	27.0
4	6775888955	1175.52	32.0
5	1844505072	922.25	13.0
6	4057192912	689.52	18.0
7	4445114986	673.54	10.0
8	6290855005	608.47	11.0
9	2320127002	540.87	8.0
10	5577150313	540.71	12.0

Execution finished without errors.

-- 🧠 Insight: By comparing calories burned per km and per active minute, we can spot users who burn more calories efficiently (either due to intensity or metabolic rate).

-- Users with high calories/km and low active minutes may be engaging in high-effort, short-duration exercises.

hourlyActivity_merge.csv

Created a merge file name **hourlyActivity_merge.csv** using
hourlyCalories_merged.csv hourlyIntensities_merged.csv hourlySteps_merged.csv

1. merged table created

The screenshot shows the DB Browser for SQLite interface. The title bar says "DB Browser for SQLite - C:\Users\DELL\OneDrive\Desktop\Hourlydata.db". The toolbar includes File, Edit, View, Tools, Help, New Database, Open Database, Write Changes, Revert Changes, Undo, Open Project, Save Project, Attach Database, and Close Database. Below the toolbar are tabs for Database Structure, Browse Data, Edit Pragmas, and Execute SQL. The main area has a tab labeled "SQL 1*". The SQL code creates the table and performs three joins:

```
1 CREATE TABLE hourlyActivity_merged AS
2   SELECT
3     c.Id,
4     c.ActivityHour,
5     s.StepTotal,
6     i.TotalIntensity,
7     i.AverageIntensity,
8     c.Calories
9   FROM hourlyCalories_merged c
10  LEFT JOIN hourlysteps_merged s
11    ON c.Id = s.Id AND c.ActivityHour = s.ActivityHour
12  LEFT JOIN hourlyIntensities_merged i
13    ON c.Id = i.Id AND c.ActivityHour = i.ActivityHour;
14
15  SELECT * FROM hourlyActivity_merged ;
```

Below the SQL code is a table showing the merged data:

	Id	ActivityHour	StepTotal	TotalIntensity	AverageIntensity	Calories
1	1503960366	4/12/2016 12:00:00 AM	373	20	0.333333	81
2	1503960366	4/12/2016 1:00:00 AM	160	8	0.133333	61
3	1503960366	4/12/2016 2:00:00 AM	151	7	0.116667	59
4	1503960366	4/12/2016 3:00:00 AM	0	0	0.0	47
5	1503960366	4/12/2016 4:00:00 AM	0	0	0.0	48
6	1503960366	4/12/2016 5:00:00 AM	0	0	0.0	48
7	1503960366	4/12/2016 6:00:00 AM	0	0	0.0	48
8	1503960366	4/12/2016 7:00:00 AM	0	0	0.0	47
9	1503960366	4/12/2016 8:00:00 AM	250	13	0.216667	68
10	1503960366	4/12/2016 9:00:00 AM	1864	30	0.5	141

Execution finished without errors.
Result: 22099 rows returned in 17ms
At line 1:
-- CREATE TABLE hourlyActivity_merged AS
-- SELECT
-- c.Id,

2. Data Cleaning and pre-processing

The screenshot shows the DB Browser for SQLite interface. The title bar says "DB Browser for SQLite - C:\Users\DELL\OneDrive\Desktop\Hourlydata.db". The toolbar and tabs are the same as in the previous screenshot. The main area has a tab labeled "SQL 2*". The SQL code performs data cleaning by summing up null values for each column:

```
18 --2. Data Cleaning and pre processing
19
20 --Check for Missing or Null Values
21
22   SELECT
23     SUM(CASE WHEN Id IS NULL THEN 1 ELSE 0 END) AS NullId,
24     SUM(CASE WHEN ActivityHour IS NULL THEN 1 ELSE 0 END) AS NullActivityHour,
25     SUM(CASE WHEN StepTotal IS NULL THEN 1 ELSE 0 END) AS NullStepTotal,
26     SUM(CASE WHEN TotalIntensity IS NULL THEN 1 ELSE 0 END) AS NullTotalIntensity,
27     SUM(CASE WHEN AverageIntensity IS NULL THEN 1 ELSE 0 END) AS NullAverageIntensity,
28     SUM(CASE WHEN Calories IS NULL THEN 1 ELSE 0 END) AS NullCalories
29   FROM hourlyActivity_merged;
```

Below the SQL code is a table showing the results of the data cleaning:

	NullId	NullActivityHour	NullStepTotal	NullTotalIntensity	NullAverageIntensity	NullCalories
1	0	0	0	0	0	0

Execution finished without errors.
Result: 1 rows returned in 29ms
At line 16:
--
-- Data Cleaning and pre processing
--Check for Missing or Null Values
SELECT
 SUM(CASE WHEN Id IS NULL THEN 1 ELSE 0 END) AS NullId,
 SUM(CASE WHEN ActivityHour IS NULL THEN 1 ELSE 0 END) AS NullActivityHour,
 SUM(CASE WHEN StepTotal IS NULL THEN 1 ELSE 0 END) AS NullStepTotal,
 SUM(CASE WHEN TotalIntensity IS NULL THEN 1 ELSE 0 END) AS NullTotalIntensity,
 SUM(CASE WHEN AverageIntensity IS NULL THEN 1 ELSE 0 END) AS NullAverageIntensity,
 SUM(CASE WHEN Calories IS NULL THEN 1 ELSE 0 END) AS NullCalories
FROM hourlyActivity_merged;

```

33  -- Check for Duplicates
34  SELECT
35    Id, ActivityHour, COUNT(*) AS Count
36  FROM hourlyActivity_merged
37  GROUP BY Id, ActivityHour
38  HAVING Count > 1;
39

```

Id | ActivityHour | Count

Execution finished without errors.
Result: 0 rows returned in 33ms
At line 33:
-- Check for Duplicates
SELECT
Id, ActivityHour, COUNT(*) AS Count
FROM hourlyActivity_merged
GROUP BY Id, ActivityHour
HAVING Count > 1;

```

40  --Standardize Date Format (Optional, for easier filtering)
41  SELECT DISTINCT ActivityHour
42  FROM hourlyActivity_merged
43  LIMIT 5;
44
45  --Steps or Calories Negative?
46  SELECT *
47  FROM hourlyActivity_merged
48  WHERE StepTotal < 0 OR Calories < 0;
49

```

Id | ActivityHour | StepTotal | TotalIntensity | AverageIntensity | Calories

Execution finished without errors.
Result: 0 rows returned in 9ms
At line 45:
--Steps or Calories Negative?
SELECT *
FROM hourlyActivity_merged
WHERE StepTotal < 0 OR Calories < 0;

3. Data Analytics & insights

```

60  |--3. Data Analysis & insights
61  |   |-- Total Steps, Calories, and Intensity by User
62
63  |   SELECT
64    |     Id,
65    |     SUM(StepTotal) AS TotalSteps,
66    |     SUM(Calories) AS TotalCalories,
67    |     SUM(TotalIntensity) AS TotalIntensity
68  |   FROM hourlyActivity_merged
69  |   GROUP BY Id
70  |   ORDER BY TotalSteps DESC;
71  |   -- Insight: Identifies the most active users.
72

```

	Id	TotalSteps	TotalCalories	TotalIntensity
1	8877689391	495623	105746	14025
2	8053475328	457463	91310	13193
3	1503960366	374546	56287	11594
4	2022484408	351712	77633	12535
5	3977333714	328244	45302	10599
6	4388161847	319765	94167	10519

Execution finished without errors.
Result: 33 rows returned in 31ms
At line 60:
--3. Data Analysis & insights
-- Total Steps, Calories, and Intensity by User
SELECT
Id,
SUM(StepTotal) AS TotalSteps,
SUM(Calories) AS TotalCalories,
SUM(TotalIntensity) AS TotalIntensity
FROM hourlyActivity_merged
GROUP BY Id
ORDER BY TotalSteps DESC;

```

74    -- Average Hourly Activity Across All Users
75
76    SELECT
77        strftime('%H', ActivityHour) AS Hour,
78        ROUND(AVG(StepTotal), 2) AS AvgSteps,
79        ROUND(AVG(Calories), 2) AS AvgCalories,
80        ROUND(AVG(TotalIntensity), 2) AS AvgIntensity
81    FROM hourlyactivity_merged
82    GROUP BY Hour
83    ORDER BY Hour;
84
85    -- 🌟 Insight: Shows which hours of the day are most active across all users. Useful for Bellabeat to suggest
86    -- optimal workout times.
87
```

Hour	AvgSteps	AvgCalories	AvgIntensity
1	NULL	320.17	97.39
		12.04	

Execution finished without errors.
Result: 1 rows returned in 32ms
At line 71:
-- Insight: Identifies the most active users.

```

-- Average Hourly Activity Across All Users
SELECT
    strftime('%H', ActivityHour) AS Hour,
    ROUND(AVG(StepTotal), 2) AS AvgSteps,
    ROUND(AVG(Calories), 2) AS AvgCalories,
    ROUND(AVG(TotalIntensity), 2) AS AvgIntensity
FROM hourlyactivity_merged
GROUP BY Hour
ORDER BY Hour;
```

```

91    -- Top 5 Most Active Hours for a Specific User
92    SELECT
93        ActivityHour,
94        StepTotal,
95        Calories,
96        TotalIntensity
97    FROM hourlyactivity_merged
98    WHERE Id = 1503960366
99    ORDER BY StepTotal DESC
100   LIMIT 5;
101   -- 🌟 Insight: Helps understand user behavior at peak performance times.
102
```

	ActivityHour	StepTotal	Calories	TotalIntensity
1	4/22/2016 9:00:00 PM	5934	311	159
2	4/27/2016 7:00:00 PM	5786	318	150
3	4/25/2016 9:00:00 PM	4895	267	132
4	4/24/2016 7:00:00 PM	4871	270	131
5	5/6/2016 6:00:00 PM	4472	261	115

At line 85:
-- 🌟 Insight: Shows which hours of the day are most active across all users. Useful for Bellabeat to suggest
-- optimal workout times.

```

-- Top 5 Most Active Hours for a Specific User
SELECT
    ActivityHour,
    StepTotal,
    Calories,
    TotalIntensity
FROM hourlyactivity_merged
WHERE Id = 1503960366
ORDER BY StepTotal DESC
LIMIT 5;
```

```

104    -- Sedentary Hours (Zero Steps)
105    SELECT COUNT(*) AS SedentaryHours
106    FROM hourlyactivity_merged
107    WHERE StepTotal = 0;
108    -- 🌟 Insight: Shows how often users are inactive - Bellabeat could encourage movement reminders.
109
110
```

SedentaryHours
1 9297

Execution finished without errors.
Result: 1 rows returned in 9ms
At line 101:
-- 🌟 Insight: Helps understand user behavior at peak performance times.

```

-- Sedentary Hours (Zero Steps)
SELECT COUNT(*) AS SedentaryHours
FROM hourlyactivity_merged
WHERE StepTotal = 0;
```

```

112    -- Active vs Inactive Hours Distribution
113    SELECT
114        CASE
115            WHEN StepTotal = 0 THEN 'Inactive'
116            ELSE 'Active'
117        END AS ActivityStatus,
118        COUNT(*) AS HourCount
119    FROM hourlyActivity_merged
120    GROUP BY ActivityStatus;
121    -- 🌟 Insight: Bellabeat can tailor health prompts or coaching messages around inactivity.

```

ActivityStatus	HourCount
1 Active	12802
2 Inactive	9297

Execution finished without errors.
 Result: 2 rows returned in 22ms
 At line 108:
 --- 🌟 Insight: Shows how often users are inactive - Bellabeat could encourage movement reminders.

```

-- Active vs Inactive Hours Distribution
SELECT
    CASE
        WHEN StepTotal = 0 THEN 'Inactive'
        ELSE 'Active'
    END AS ActivityStatus,
    COUNT(*) AS HourCount
FROM hourlyActivity_merged
GROUP BY ActivityStatus;

```

```

125    --Calories Burned by Hour Across All Users
126    SELECT
127        strftime('%H', ActivityHour) AS Hour,
128        ROUND(SUM(Calories), 2) AS TotalCaloriesBurned
129    FROM hourlyActivity_merged
130    GROUP BY Hour
131    ORDER BY TotalCaloriesBurned DESC;
132    -- 🌟 Insight: Ideal for plotting a heatmap or bar chart in Power BI to visualize energy expenditure patterns.

```

Hour	TotalCaloriesBurned
1 NULL	2152150.0

```

135    -- Users with Consistently High Hourly Steps (>500 avg/hour)
136    SELECT
137        Id,
138        ROUND(AVG(StepTotal), 2) AS AvgStepsPerHour
139    FROM hourlyActivity_merged
140    GROUP BY Id
141    HAVING AvgStepsPerHour > 500
142    ORDER BY AvgStepsPerHour DESC;
143    -- 🌟 Insight: Target high-performance users for premium product marketing or gamification.

```

Id	AvgStepsPerHour
1 8877689391	674.32
2 8053475328	622.4
3 1503960366	522.38