

SQL QUERIES & RESULT:

Table 1: dailyActivity

Query 1:

```
PRAGMA table_info(dailyActivity);
```

Result 1:

0	Id	INTEGER	0	0
1	ActivityDate	TEXT	0	0
2	TotalSteps	INTEGER	0	0
3	TotalDistance	REAL	0	0
4	TrackerDistance	REAL	0	0
5	LoggedActivitiesDistance	INTEGER	0	0
6	VeryActiveDistance	REAL	0	0
7	ModeratelyActiveDistance	REAL	0	0
8	LightActiveDistance	REAL	0	0
9	SedentaryActiveDistance	INTEGER	0	0
10	VeryActiveMinutes	INTEGER	0	0
11	FairlyActiveMinutes	INTEGER	0	0
12	LightlyActiveMinutes	INTEGER	0	0
13	SedentaryMinutes	INTEGER	0	0
14	Calories	INTEGER	0	0

Explanation 1:

- This query displays the structure of the **“dailyActivity”** table, including column names, data types, and constraints.
- It helps understand what kind of data is stored in the table before performing any analysis.

Query 2:

```
SELECT * FROM dailyActivity LIMIT 5;
```

Result 2:

1503960366	4/12/2016	13162	8.5	8.5	0	1.88	0.55	6.06	0	25	13	328	728	1985
1503960366	4/13/2016	10735	6.97	6.97	0	1.57	0.69	4.71	0	21	19	217	776	1797
1503960366	4/14/2016	10460	6.74	6.74	0	2.44	0.4	3.91	0	30	11	181	1218	1776
1503960366	4/15/2016	9762	6.28	6.28	0	2.14	1.26	2.83	0	29	34	209	726	1745
1503960366	4/16/2016	12669	8.16	8.16	0	2.71	0.41	5.04	0	36	10	221	773	1863

Explanation 2:

- This query retrieves the first 5 rows from the “**dailyActivity**” table.
- It is useful for quickly previewing the data and understanding the table’s contents and format.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM dailyActivity;
```

Result 3:

	total_rows
1	940

Explanation 3:

- This query counts the total number of records (rows) present in the “**dailyActivity**” table.
- It helps understand the dataset size, which is useful for planning data analysis or filtering.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM dailyActivity;
```

Result 4:

	unique_participants
1	33

Explanation 4:

- This query returns the number of unique participants by counting distinct Id values in the “**dailyActivity**” table.
- It helps identify how many individual participants contributed data to the dataset.

Query 5:

SELECT

MIN(ActivityDate) AS earliest_date,

MAX(ActivityDate) AS latest_date

FROM dailyActivity;

Result 5:

	earliest_date	latest_date
1	4/12/2016	5/9/2016

Explanation 5:

- This query finds the earliest and latest activity dates in the “**dailyActivity**” table.
- It helps determine the date range covered by the dataset for time-based analysis.

Query 6:

SELECT

MIN(TotalSteps) AS min_steps,

MAX(TotalSteps) AS max_steps

FROM dailyActivity;

Result 6:

	min_steps	max_steps
1	0	36019

Explanation 6:

- This query retrieves the minimum and maximum number of steps recorded by users in the dataset.
- It helps understand the range of steps levels among participants.

Query 7:

SELECT

MIN(TotalDistance) AS min_distance,

MAX(TotalDistance) AS max_distance

FROM dailyActivity;

Result 7:

	min_distance	max_distance
1	0.0	28.0300006866455

Explanation 7:

- This query returns the shortest and longest total distances recorded in the “**dailyActivity**” table.
- It helps identify the range of distance covered by users during their activities.

Query 8:

SELECT

MIN(VeryActiveMinutes) AS min_very_active_minutes,

MAX(VeryActiveMinutes) AS max_very_active_minutes

FROM dailyActivity;

Result 8:

	min_very_active_minutes	max_very_active_minutes
1	0	210

Explanation 8:

- This SQL query retrieves the minimum and maximum values of the **Very Active Minutes** column from the “**dailyActivity**” table.
- It helps understand the lowest and highest amount of very active minutes tracked in the dataset.

Query 9:

SELECT

MIN(FairlyActiveMinutes) AS min_fairly_active_minutes,

MAX(FairlyActiveMinutes) AS max_fairly_active_minutes

FROM dailyActivity;

Result 9:

	min_fairly_active_minutes	max_fairly_active_minutes
1	0	143

Explanation 9:

- This query returns the minimum and maximum values of Fairly Active Minutes from the “**dailyActivity**” table.
- It helps identify the range of moderately active minutes recorded by users in the dataset.

Query 10:

SELECT

MIN(LightlyActiveMinutes) AS min_lightly_active_minutes,

MAX(LightlyActiveMinutes) AS max_lightly_active_minutes

FROM dailyActivity;

Result 10:

	min_lightly_active_minutes	max_lightly_active_minutes
1	0	518

Explanation 10:

- This query retrieves the minimum and maximum values of Lightly Active Minutes from the “dailyActivity” table.
- It provides insight into the range of light physical activity durations recorded by users.

Query 11:

SELECT

MIN(SedentaryMinutes) AS min_sedentary_minutes,

MAX(SedentaryMinutes) AS max_sedentary_minutes

FROM dailyActivity;

Result 11:

	min_sedentary_minutes	max_sedentary_minutes
1	0	1440

Explanation 11:

- This SQL query extracts the minimum and maximum values of Sedentary Minutes from the “dailyActivity” table.
- It shows the least and most time users spent being sedentary in a day, useful for assessing inactivity patterns.

Query 12:

SELECT

MIN(Calories) AS min_calories,

MAX(Calories) AS max_calories

FROM dailyActivity;

Result 12:

	min_calories	max_calories
1	0	4900

Explanation 12:

- This query retrieves the minimum and maximum number of Calories burned from the “**dailyActivity**” table.
- It helps determine the calorie burn range among users, useful for analyzing energy expenditure patterns.

Query 13:

SELECT

COUNT(*) AS records_with_nulls

FROM dailyActivity

WHERE

Id IS NULL OR

ActivityDate IS NULL OR

TotalSteps IS NULL OR

Calories IS NULL;

Result 13:

	records_with_nulls
1	0

Explanation 13:

- This query counts the number of records in the “**dailyActivity**” table that have NULL values in any of the columns: Id, ActivityDate, TotalSteps, or Calories.
- It helps identify incomplete or missing data entries that may need cleaning or handling before analysis.

Table 2: dailyCalories

Query 1:

```
PRAGMA table_info(dailyCalories);
```

Result 1:

	cid	name	type	notnull	dflt_value	pk
1	0	Id	INTEGER	0	NULL	0
2	1	ActivityDay	TEXT	0	NULL	0
3	2	Calories	INTEGER	0	NULL	0

Explanation 1:

- The query PRAGMA returns the schema information of the “**dailyCalories**” table.
- It displays details like column names, data types, whether a column can be NULL, default values.

Query 2:

```
SELECT * FROM dailyCalories LIMIT 5;
```

Result 2:

	Id	ActivityDay	Calories
1	1503960366	4/12/2016	1985
2	1503960366	4/13/2016	1797
3	1503960366	4/14/2016	1776
4	1503960366	4/15/2016	1745
5	1503960366	4/16/2016	1863

Explanation 2:

- The query retrieves all columns and the first 5 rows from the “**dailyCalories**” table.
- It provides a quick preview of the table's data structure and sample records.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM dailyCalories;
```

Result 3:

	total_rows
1	940

Explanation 3:

- The query counts and returns the total number of rows in the dailyCalories table, labeled as total_rows.
- It helps assess the dataset's size

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM dailyCalories;
```

Result 4:

	unique_participants
1	33

Explanation 4:

- The query counts and returns the number of unique Id values in the “dailyCalories” table.
- It helps determine how many distinct individuals are represented in the dataset.

Query 5:

```
SELECT
```

```
    MIN(ActivityDay) AS earliest_date,
```

```
    MAX(ActivityDay) AS latest_date
```

```
FROM dailyCalories;
```

Result 5:

	earliest_date	latest_date
1	4/12/2016	5/9/2016

Explanation 5:

- The query finds the earliest (minimum) and latest (maximum) dates in the ActivityDay.
- It helps identify the time range covered by the dataset.

Query 6:

SELECT

MIN(Calories) AS min_calories,

MAX(Calories) AS max_calories

FROM dailyCalories;

Result 6:

	min_calories	max_calories
1	0	4900

Explanation 6:

- The query identifies the lowest (MIN) and highest (MAX) calorie values.
- This helps understand the range of calorie expenditure in the dataset.

Query 7:

SELECT

COUNT(*) AS records_with_nulls

FROM dailyCalories

WHERE

Id IS NULL OR

ActivityDay IS NULL OR

Calories IS NULL;

Result 7:

records_with_nulls	
1	0

Explanation 7:

- The query counts how many rows in the dailyCalories table contain NULL values in any of the specified columns (Id, ActivityDay, or Calories).
- This helps assess data completeness by identifying missing or incomplete records in the dataset.

Table 3: dailyIntensities

Query 1:

```
PRAGMA table_info(dailyIntensities);
```

Result 1:

	cid	name	type	notnull	dflt_value	pk
1	0	Id	INTEGER	0	NULL	0
2	1	ActivityDay	TEXT	0	NULL	0
3	2	SedentaryMinutes	INTEGER	0	NULL	0
4	3	LightlyActiveMinutes	INTEGER	0	NULL	0
5	4	FairlyActiveMinutes	INTEGER	0	NULL	0
6	5	VeryActiveMinutes	INTEGER	0	NULL	0
7	6	SedentaryActiveDistance	INTEGER	0	NULL	0
8	7	LightActiveDistance	REAL	0	NULL	0
9	8	ModeratelyActiveDistance	REAL	0	NULL	0
10	9	VeryActiveDistance	REAL	0	NULL	0

Explanation 1:

- The query PRAGMA returns the schema information of the “dailyIntensities” table.
- It provides a schema overview without displaying actual table data, useful for understanding the table's design.

Query 2:

```
SELECT * FROM dailyIntensities LIMIT 5;
```

Result 2:

Id	ActivityDay	SedentaryMinutes	LightlyActiveMinutes	FairlyActiveMinutes	VeryActiveMinutes	SedentaryActiveDistance	LightlyActiveDistance	ModeratelyActiveDistance	VeryActiveDistance
1503960366	4/12/2016	728	328	13	25	0	6.059999994277954	0.550000011920929	1.87999999523163
1503960366	4/13/2016	776	217	19	21	0	4.71000003814697	0.689999997615814	1.57000005245209
1503960366	4/14/2016	1218	181	11	30	0	3.91000008583069	0.400000005960464	2.44000005722046
1503960366	4/15/2016	726	209	34	29	0	2.829999992370605	1.25999999046326	2.14000010490417
1503960366	4/16/2016	773	221	10	36	0	5.03999996185303	0.409999996423721	2.71000003814697

Explanation 2:

- The query retrieves all columns and the first 5 rows from the dailyIntensities table.
- This helps verify the table's contents and column layout before deeper analysis.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM dailyIntensities;
```

Result 3:

total_rows
940

Explanation 3:

- The query counts and returns the total number of rows in the dailyIntensities table.
- This helps assess the scale of the data before performing further analysis or processing.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM dailyIntensities;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts and returns the number of unique participant IDs (Id) in the dailyIntensities table.
- This helps determine how many distinct individuals are included in the dataset, ensuring no duplicates in participant tracking.

Query 5:

```
SELECT  
  
    MIN(ActivityDay) AS earliest_date,  
  
    MAX(ActivityDay) AS latest_date  
  
FROM dailyIntensities;
```

Result 5:

earliest_date	latest_date
4/12/2016	5/9/2016

Explanation 5:

- The query identifies the first (earliest) and last (latest) dates recorded in the ActivityDay column.
- This reveals the time covered by the dataset.

Query 6:

```
SELECT  
  
    MIN(SedentaryMinutes) AS min_sedentary_minutes,  
  
    MAX(SedentaryMinutes) AS max_sedentary_minutes  
  
FROM dailyIntensities;
```

Result 6:

min_sedentary_minutes	max_sedentary_minutes
0	1440

Explanation 6:

- The query calculates the minimum and maximum values in the SedentaryMinutes column, showing the range of inactive time recorded.
- This helps identify potential outliers.

Query 7:

```
SELECT  
  
    MIN(LightlyActiveMinutes) AS min_lightly_active_minutes,  
  
    MAX(LightlyActiveMinutes) AS max_lightly_active_minutes  
  
FROM dailyIntensities;
```

Result 7:

min_lightly_active_minutes	max_lightly_active_minutes
0	518

Explanation 7:

- The query finds the smallest and largest values in the LightlyActiveMinutes column, showing the range of low-intensity activity recorded per day.
- This helps assess activity patterns.

Query 8:

```
SELECT  
  
    MIN(FairlyActiveMinutes) AS min_fairly_active_minutes,  
  
    MAX(FairlyActiveMinutes) AS max_fairly_active_minutes  
  
FROM dailyIntensities;
```

Result 8:

min_fairly_active_minutes	max_fairly_active_minutes
0	143

Explanation 8:

- The query retrieves the minimum and maximum values from the FairlyActiveMinutes column, indicating the range of moderate-intensity activity.
- This reveals how much users engage in sustained, effortful movement.

Query 9:

```
SELECT  
  
    MIN(VeryActiveMinutes) AS min_very_active_minutes,  
  
    MAX(VeryActiveMinutes) AS max_very_active_minutes  
  
FROM dailyIntensities;
```

Result 9:

min_very_active_minutes	max_very_active_minutes
0	210

Explanation 9:

- The query extracts the lowest and highest values from the VeryActiveMinutes column, showing the range of vigorous activity.
- A high max value could indicate athletes, while 0 mins reveals completely sedentary days.

Query 10:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM dailyIntensities  
  
WHERE  
  
    Id IS NULL OR  
  
    ActivityDay IS NULL OR  
  
    SedentaryMinutes IS NULL;
```

Result 10:

records_with_nulls
0

Explanation 10:

- The query counts rows in the dailyIntensities table where critical columns (Id, ActivityDay, or SedentaryMinutes) contain NULL/missing values, labeled as records_with_nulls.
- A result > 0 flags data quality issues.

Table 4: dailySteps

Query 1:

```
PRAGMA table_info(dailySteps);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityDay	TEXT	0		0
2	StepTotal	INTEGER	0		0

Explanation 1:

- The query retrieves structural about the dailySteps table.
- It provides a schema of the table without displaying actual data.

Query 2:

```
SELECT * FROM dailySteps LIMIT 5;
```

Result 2:

Id	ActivityDay	StepTotal
1503960366	4/12/2016	13162
1503960366	4/13/2016	10735
1503960366	4/14/2016	10460
1503960366	4/15/2016	9762
1503960366	4/16/2016	12669

Explanation 2:

- The query retrieves all columns and the first 5 rows from the dailySteps table.
- Helps verify data integrity and column order.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM dailySteps;
```

Result 3:

total_rows
940

Explanation 3:

- The query counts and returns the total number of records in the dailySteps table.
- This helps assess the dataset size.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM dailySteps;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts and returns the number of unique user IDs (Id) in the dailySteps table.
- This reveals how many distinct individuals contributed step data.

Query 5:

```
SELECT  
  
    MIN(ActivityDay) AS earliest_date,  
  
    MAX(ActivityDay) AS latest_date  
  
FROM dailySteps;
```

Result 5:

earliest_date	latest_date
4/12/2016	5/9/2016

Explanation 5:

- The query identifies the oldest (MIN) and most recent (MAX) dates in the ActivityDay column.
- Helps determine if the dataset covers the expected period.

Query 6:

```
SELECT  
  
    MIN(StepTotal) AS min_steps_total,  
  
    MAX(StepTotal) AS max_steps_total  
  
FROM dailySteps;
```

Result 6:

min_steps_total	max_steps_total
0	36019

Explanation 6:

- The query identifies the lowest (0 steps) and highest (36K steps) daily step counts.
- Helps flag inactive days or outliers.

Query 7:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM dailySteps  
  
WHERE  
  
    Id IS NULL OR  
  
    ActivityDay IS NULL OR  
  
    StepTotal IS NULL;
```

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows in the dailySteps table with missing values in columns.
- A result > 0 indicates data quality issues.

Table 5: heartrate_Seconds

Query 1:

```
PRAGMA table_info(heartrate_Seconds);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	Time	TEXT	0		0
2	Value	INTEGER	0		0

Explanation 1:

- The query retrieves the schema details of the heartrate_seconds table.
- It helps understand the structure of heart rate data.

Query 2:

```
SELECT * FROM heartrate_Seconds LIMIT 5;
```

Result 2:

Id	Time	Value
2022484408	4/12/2016 7:21:00 AM	97
2022484408	4/12/2016 7:21:05 AM	102
2022484408	4/12/2016 7:21:10 AM	105
2022484408	4/12/2016 7:21:20 AM	103
2022484408	4/12/2016 7:21:25 AM	101

Explanation 2:

- The query retrieves all columns and the first 5 rows from the heartrate_seconds table.
- Helps verify data format and typical values.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM heartrate_Seconds;
```

Result 3:

total_rows
2483658

Explanation 3:

- The query counts and returns the total number of heart rate readings in the heartrate_seconds table,
- Useful for assessing data volume for analysis.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM heartrate_Seconds;
```

Result 4:

unique_participants
14

Explanation 4:

- The query counts the number of unique users (Id) in the heartrate_seconds table.
- Helps validate participant coverage and identify potential gaps in data collection.

Query 5:

```
SELECT  
  
    MIN(Time) AS earliest_datetime,  
  
    MAX(Time) AS latest_datetime  
  
FROM heartrate_Seconds;
```

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:59:59 PM

Explanation 5:

- The query identifies the first and last recorded timestamps in the heartrate_seconds table.
- Useful for verifying data continuity or identifying gaps in monitoring periods.

Query 6:

```
SELECT  
  
    MIN(Value) AS min_hearttrate,  
  
    MAX(Value) AS max_hearttrate  
  
FROM heartrate_Seconds;
```

Result 6:

min_hearttrate	max_hearttrate
36	203

Explanation 6:

- The query finds the lowest and highest heart rate values recorded in the dataset, exposing potential outliers or extreme physiological states.
- Helps flag data anomalies or clinically significant events.

Query 7:

SELECT

COUNT(*) AS records_with_nulls

FROM heartrate_Seconds

WHERE

Id IS NULL OR

Time IS NULL OR

Value IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing critical data in the heartrate_seconds table.

Table 6: hourlyCalories

Query 1:

```
PRAGMA table_info(hourlyCalories);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityHour	TEXT	0		0
2	Calories	INTEGER	0		0

Explanation 1:

- The query retrieves the schema structure of the hourlyCalories table.
- Helps understand the data organization.

Query 2:

```
SELECT * FROM hourlyCalories LIMIT 5;
```

Result 2:

Id	ActivityHour	Calories
1503960366	4/12/2016 12:00:00 AM	81
1503960366	4/12/2016 1:00:00 AM	61
1503960366	4/12/2016 2:00:00 AM	59
1503960366	4/12/2016 3:00:00 AM	47
1503960366	4/12/2016 4:00:00 AM	48

Explanation 2:

- The query retrieves all columns and the first 5 rows from the hourlyCalories table.
- Helps verify data format and typical values.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM hourlyCalories;
```

Result 3:

total_rows
22099

Explanation 3:

- The query counts and returns the total number of hourly calorie records in the hourlyCalories table.
- Helps assess data volume and processing needs.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM hourlyCalories;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of distinct users (Id) in the hourlyCalories table, revealing how many individuals contributed hourly calorie data.
- Helps determine participant coverage and identify potential data gaps.

Query 5:

```
SELECT
```

```
MIN(ActivityHour) AS earliest_datetime,
```

MAX(ActivityHour) AS latest_datetime

FROM hourlyCalories

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:00:00 PM

Explanation 5:

- The query identifies the oldest (MIN) and newest (MAX) timestamps in the ActivityHour column, showing the time range of hourly calorie data.
- Helps validate temporal coverage and detect missing periods.

Query 6:

SELECT

MIN(Calories) AS min_calories,

MAX(Calories) AS max_calories

FROM hourlyCalories;

Result 6:

min_calories	max_calories
42	948

Explanation 6:

- The query finds the minimum and maximum hourly calorie burn values in the dataset, revealing the range of energy expenditure.

Query 7:

SELECT

COUNT(*) AS records_with_nulls

FROM hourlyCalories

WHERE

Id IS NULL OR

ActivityHour IS NULL OR

Calories IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing values in critical columns, flagging potential data quality issues.

Table 7: hourlyIntensities

Query 1:

```
PRAGMA table_info(hourlyIntensities);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityHour	TEXT	0		0
2	TotalIntensity	INTEGER	0		0
3	AverageIntensity	REAL	0		0

Explanation 1:

- The query retrieves schema details of the hourlyIntensities table, including column names, data types, and NULL constraints.
- Helps verify data structure before analysis or joins.

Query 2:

```
SELECT * FROM hourlyIntensities LIMIT 5;
```

Result 2:

Id	ActivityHour	TotalIntensity	AverageIntensity
1503960366	4/12/2016 12:00:00 AM	20	0.333333
1503960366	4/12/2016 1:00:00 AM	8	0.133333
1503960366	4/12/2016 2:00:00 AM	7	0.116667
1503960366	4/12/2016 3:00:00 AM	0	0.0
1503960366	4/12/2016 4:00:00 AM	0	0.0

Explanation 2:

- The query fetches all columns and the first 5 rows from the hourlyIntensities table.
- Useful for verifying data format and sample values.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM hourlyIntensities;
```

Result 3:

total_rows
22099

Explanation 3:

- The query counts and returns the total number of hourly intensity records in the hourlyIntensities table.
- Helps assess data volume and processing requirements.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM hourlyIntensities;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of unique users (Id) in the hourlyIntensities table.
- Helps confirm data consistency with other hourly tables.

Query 5:

SELECT

MIN(ActivityHour) AS earliest_datetime,

MAX(ActivityHour) AS latest_datetime

FROM hourlyIntensities;

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:00:00 PM

Explanation 5:

- The query identifies the time range of the dataset by extracting the earliest (MIN) and latest (MAX) timestamps from the ActivityHour column.
- Helps verify temporal alignment with other tables.

Query 6:

SELECT

MIN(TotalIntensity) AS min_total_intensity,

MAX(TotalIntensity) AS max_total_intensity,

MIN(AverageIntensity) AS min_average_intensity,

MAX(AverageIntensity) AS max_average_intensity

FROM hourlyIntensities;

Result 6:

min_total_intensity	max_total_intensity	min_average_intensity	max_average_intensity
0	180	0.0	3.0

Explanation 6:

- The query calculates the minimum and maximum values for both TotalIntensity and AverageIntensity in the hourlyIntensities table.

Query 7:

SELECT

COUNT(*) AS records_with_nulls

FROM hourlyIntensities

WHERE

Id IS NULL OR

ActivityHour IS NULL OR

TotalIntensity IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows in the hourlyIntensities table with NULL values in critical columns.

Table 8: hourlySteps

Query 1:

```
PRAGMA table_info(hourlySteps);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityHour	TEXT	0		0
2	StepTotal	INTEGER	0		0

Explanation 1:

- The query retrieves schema metadata for the hourlySteps table.
- Essential for understanding the table's structure before querying or joining with other datasets.

Query 2:

```
SELECT * FROM hourlySteps LIMIT 5;
```

Result 2:

Id	ActivityHour	StepTotal
1503960366	4/12/2016 12:00:00 AM	373
1503960366	4/12/2016 1:00:00 AM	160
1503960366	4/12/2016 2:00:00 AM	151
1503960366	4/12/2016 3:00:00 AM	0
1503960366	4/12/2016 4:00:00 AM	0

Explanation 2:

- The query retrieves the first 5 rows and all columns from the hourlySteps table.
- Helps verify data structure and spot-check for anomalies.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM hourlySteps;
```

Result 3:

total_rows
22099

Explanation 3:

- The query counts and returns the total number of hourly step records in the hourlySteps table,
- Helps assess data volume and consistency with related tables.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM hourlySteps;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of distinct users (Id) in the hourlySteps table.
- Helps verify participant coverage and ensures consistency with other hourly tables.

Query 5:

```
SELECT  
    MIN(ActivityHour) AS earliest_datetime,  
    MAX(ActivityHour) AS latest_datetime  
FROM hourlySteps;
```

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:00:00 PM

Explanation 5:

- The query identifies the oldest and newest timestamps in the ActivityHour column.
- Confirms temporal alignment with other hourly datasets.

Query 6:

```
SELECT  
  
    MIN(StepTotal) AS min_step_total,  
  
    MAX(StepTotal) AS max_step_total  
  
FROM hourlySteps;
```

Result 6:

min_step_total	max_step_total
0	10554

Explanation 6:

- The query calculates the minimum and maximum hourly step counts in the dataset.

Query 7:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM hourlySteps
```

WHERE

Id IS NULL OR

ActivityHour IS NULL OR

StepTotal IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing values in critical columns.

Table 9: minuteCaloriesNarrow

Query 1:

```
PRAGMA table_info(minuteCaloriesNarrow);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityMinute	TEXT	0		0
2	Calories	REAL	0		0

Explanation 1:

- The query retrieves schema details of the minuteCaloriesNarrow table.
- Helps understand the structure of per-minute calorie data.

Query 2:

```
SELECT * FROM minuteCaloriesNarrow LIMIT 5;
```

Result 2:

Id	ActivityMinute	Calories
1503960366	4/12/2016 12:00:00 AM	0.786499977111816
1503960366	4/12/2016 12:01:00 AM	0.786499977111816
1503960366	4/12/2016 12:02:00 AM	0.786499977111816
1503960366	4/12/2016 12:03:00 AM	0.786499977111816
1503960366	4/12/2016 12:04:00 AM	0.786499977111816

Explanation 2:

- The query retrieves the first 5 minute-by-minute calorie records.
- Helps validate data granularity and spot-check for anomalies.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM minuteCaloriesNarrow;
```

Result 3:

total_rows
1325580

Explanation 3:

- The query counts and returns the total number of minute-level calorie records in the minuteCaloriesNarrow table.
- Helps assess computational load for analysis.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteCaloriesNarrow;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of unique users (Id) in the minuteCaloriesNarrow table.
- Helps verify participant coverage and consistency with other tables.

Query 5:

SELECT

MIN(ActivityMinute) AS earliest_datetime,

MAX(ActivityMinute) AS latest_datetime

FROM minuteCaloriesNarrow;

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:59:00 PM

Explanation 5:

- The query identifies the first and last recorded timestamps in the ActivityMinute column.
- Helps confirm temporal alignment with other minute-level datasets.

Query 6:

SELECT

MIN(Calories) AS min_calories,

MAX(Calories) AS max_calories

FROM minuteCaloriesNarrow;

Result 6:

min_calories	max_calories
0.0	19.7499465942383

Explanation 6:

- The query finds the lowest and highest calorie values recorded per minute, exposing the range of energy expenditure.

Query 7:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM minuteCaloriesNarrow  
  
WHERE  
  
    Id IS NULL OR  
  
    ActivityMinute IS NULL OR  
  
    Calories IS NULL;
```

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing values in critical columns.

Table 10: minuteCaloriesWide

Query 1:

```
PRAGMA table_info(minuteCaloriesWide);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityHour	TEXT	0		0
2	Calories00	REAL	0		0
3	Calories01	REAL	0		0
4	Calories02	REAL	0		0
5	Calories03	REAL	0		0
6	Calories04	REAL	0		0
7	Calories05	REAL	0		0
8	Calories06	REAL	0		0
9	Calories07	REAL	0		0
10	Calories08	REAL	0		0
11	Calories09	REAL	0		0
12	Calories10	REAL	0		0
13	Calories11	REAL	0		0
14	Calories12	REAL	0		0
15	Calories13	REAL	0		0
16	Calories14	REAL	0		0
17	Calories15	REAL	0		0
18	Calories16	REAL	0		0
19	Calories17	REAL	0		0
20	Calories18	REAL	0		0
21	Calories19	REAL	0		0
22	Calories20	REAL	0		0
23	Calories21	REAL	0		0

24	Calories22	REAL	0		0
25	Calories23	REAL	0		0
26	Calories24	REAL	0		0
27	Calories25	REAL	0		0
28	Calories26	REAL	0		0
29	Calories27	REAL	0		0
30	Calories28	REAL	0		0
31	Calories29	REAL	0		0
32	Calories30	REAL	0		0
33	Calories31	REAL	0		0
34	Calories32	REAL	0		0
35	Calories33	REAL	0		0
36	Calories34	REAL	0		0
37	Calories35	REAL	0		0
38	Calories36	REAL	0		0
39	Calories37	REAL	0		0
40	Calories38	REAL	0		0
41	Calories39	REAL	0		0
42	Calories40	REAL	0		0
43	Calories41	REAL	0		0
44	Calories42	REAL	0		0
45	Calories43	REAL	0		0
46	Calories44	REAL	0		0
47	Calories45	REAL	0		0
48	Calories46	REAL	0		0
49	Calories47	REAL	0		0
50	Calories48	REAL	0		0
51	Calories49	REAL	0		0
52	Calories50	REAL	0		0
53	Calories51	REAL	0		0

03 66	6 1: 00 :0 0 AM	1181 6	1181 6	1181 6	1181 6	9725 3418	9725 3418	9725 3418	1181 6	9725 3418	1181 6	9725 3418	1181 6	9725 3418	1181 6	1181 6	1181 6	1181 6	1181 6	1181 6	1181 6	1181 6
15 03 96 03 66	4/ 13 /2 01 6 2: 00 :0 0 AM	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	
15 03 96 03 66	4/ 13 /2 01 6 3: 00 :0 0 AM	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	2.04 4899 9404 9072	0.94 3799 9725 3418	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.94 3799 9725 3418	0.78 6499 9771 1181 6	0.94 3799 9725 3418	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	
15 03 96 03 66	4/ 13 /2 01 6 4: 00 :0 0 AM	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	0.94 3799 9725 3418	0.78 6499 9771 1181 6	0.78 6499 9771 1181 6	

Explanation 2:

- The query retrieves the first 5 rows from the minuteCaloriesWide table.
- Helps visualize the wide-format structure, where each row represents one hour with calories burned per minute spread across columns.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM minuteCaloriesWide;
```

Result 3:

total_rows
21645

Explanation 3:

- The query counts the total number of hourly records in the minuteCaloriesWide table.
- Helps assess data volume and consistency with related tables.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteCaloriesWide;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of distinct users (Id) in the minuteCaloriesWide table.
- Ensures consistency with other tables.

Query 5:

```
SELECT
```

```
    MIN(ActivityHour) AS earliest_datetime,
```

```
    MAX(ActivityHour) AS latest_datetime
```

```
FROM minuteCaloriesWide;
```

Result 5:

earliest_datetime	latest_datetime
4/13/2016 10:00:00 AM	5/9/2016 9:00:00 PM

Explanation 5:

- The query identifies the time range of the dataset by extracting the earliest (MIN) and latest (MAX) timestamps from the ActivityHour column.
- Helps verify temporal alignment with other tables.

Query 6:

SELECT

MIN(Calories00) AS min_cal_at_00,

MAX(Calories00) AS max_cal_at_00

FROM minuteCaloriesWide;

Result 6:

min_cal_at_00	max_cal_at_00
0.702700018882751	19.7273368835449

Explanation 6:

- The query calculates the minimum and maximum calorie values.

Query 7:

SELECT

COUNT(*) AS records_with_nulls

FROM minuteCaloriesWide

WHERE

Id IS NULL OR

ActivityHour IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing values in critical columns.

Table 11: minuteIntensitiesNarrow

Query 1:

```
PRAGMA table_info(minuteIntensitiesNarrow);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityMinute	TEXT	0		0
2	Intensity	INTEGER	0		0

Explanation 1:

- The query retrieves schema metadata for the minuteIntensitiesNarrow table.
- Helps understand the structure of minute-level intensity data.

Query 2:

```
SELECT * FROM minuteIntensitiesNarrow LIMIT 5;
```

Result 2:

1503960366	4/12/2016 12:00:00 AM	0
1503960366	4/12/2016 12:01:00 AM	0
1503960366	4/12/2016 12:02:00 AM	0
1503960366	4/12/2016 12:03:00 AM	0
1503960366	4/12/2016 12:04:00 AM	0

Explanation 2:

- The query retrieves the first 5 rows from the minuteIntensitiesNarrow table.
- Helps verify data format and sample values.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM minuteIntensitiesNarrow;
```

Result 3:

total_rows
1325580

Explanation 3:

- The query counts the total number of minute-level intensity records in the minuteIntensitiesNarrow table.
- Helps assess computational load for time-series analysis.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteIntensitiesNarrow;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of unique users (Id) in the minuteIntensitiesNarrow table.
- Ensures data consistency with other minute-level tables.

Query 5:

```
SELECT
```

```
    MIN(ActivityMinute) AS earliest_datetime,
```

MAX(ActivityMinute) AS latest_datetime

FROM minuteIntensitiesNarrow;

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:59:00 PM

Explanation 5:

- The query identifies the time range of the dataset by extracting the earliest (MIN) and latest (MAX) timestamps from the ActivityMinute column.
- Helps verify temporal alignment with other minute-level tables.

Query 6:

SELECT

MIN(Intensity) AS min_intensity,

MAX(Intensity) AS max_intensity

FROM minuteIntensitiesNarrow;

Result 6:

min_intensity	max_intensity
0	3

Explanation 6:

- The query calculates the lowest and highest intensity values in the dataset, confirming the valid range (0–3) for the Intensity column.

Query 7:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM minuteIntensitiesNarrow  
  
WHERE  
  
    Id IS NULL OR  
  
    ActivityMinute IS NULL OR  
  
    Intensity IS NULL;
```

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing values in critical columns.

Table 12: minuteIntensitiesWide

Query 1:

```
PRAGMA table_info(minuteIntensitiesWide);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityHour	TEXT	0		0
2	Intensity00	INTEGER	0		0
3	Intensity01	INTEGER	0		0
4	Intensity02	INTEGER	0		0
5	Intensity03	INTEGER	0		0
6	Intensity04	INTEGER	0		0
7	Intensity05	INTEGER	0		0
8	Intensity06	INTEGER	0		0
9	Intensity07	INTEGER	0		0
10	Intensity08	INTEGER	0		0
11	Intensity09	INTEGER	0		0
12	Intensity10	INTEGER	0		0
13	Intensity11	INTEGER	0		0
14	Intensity12	INTEGER	0		0
15	Intensity13	INTEGER	0		0
16	Intensity14	INTEGER	0		0
17	Intensity15	INTEGER	0		0
18	Intensity16	INTEGER	0		0
19	Intensity17	INTEGER	0		0
20	Intensity18	INTEGER	0		0
21	Intensity19	INTEGER	0		0
22	Intensity20	INTEGER	0		0

23	Intensity21	INTEGER	0		0
24	Intensity22	INTEGER	0		0
25	Intensity23	INTEGER	0		0
26	Intensity24	INTEGER	0		0
27	Intensity25	INTEGER	0		0
28	Intensity26	INTEGER	0		0
29	Intensity27	INTEGER	0		0
30	Intensity28	INTEGER	0		0
31	Intensity29	INTEGER	0		0
32	Intensity30	INTEGER	0		0
33	Intensity31	INTEGER	0		0
34	Intensity32	INTEGER	0		0
35	Intensity33	INTEGER	0		0
36	Intensity34	INTEGER	0		0
37	Intensity35	INTEGER	0		0
38	Intensity36	INTEGER	0		0
39	Intensity37	INTEGER	0		0
40	Intensity38	INTEGER	0		0
41	Intensity39	INTEGER	0		0
42	Intensity40	INTEGER	0		0
43	Intensity41	INTEGER	0		0
44	Intensity42	INTEGER	0		0
45	Intensity43	INTEGER	0		0
46	Intensity44	INTEGER	0		0
47	Intensity45	INTEGER	0		0
48	Intensity46	INTEGER	0		0
49	Intensity47	INTEGER	0		0
50	Intensity48	INTEGER	0		0
51	Intensity49	INTEGER	0		0
52	Intensity50	INTEGER	0		0

[illegible]

[illegible]

Explanation 2:

Query 3:

Result 3:

Explanation 3:

- The query counts the total number of hourly records in the minuteIntensitiesWide table

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteIntensitiesWide;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of unique users (Id) in the minuteIntensitiesWide table.
- Ensures data consistency with other wide-format tables.

Query 5:

```
SELECT
```

```
    MIN(ActivityHour) AS earliest_datetime,
```

```
    MAX(ActivityHour) AS latest_datetime
```

```
FROM minuteIntensitiesWide;
```

Result 5:

earliest_datetime	latest_datetime
4/13/2016 10:00:00 AM	5/9/2016 9:00:00 PM

Explanation 5:

- The query identifies the time range of the dataset by extracting the earliest (MIN) and latest (MAX) timestamps from the ActivityHour column.
- Helps verify temporal alignment with other wide-format tables.

Query 6:

SELECT

MIN(Intensity00) AS min_intensity_at_00,

MAX(Intensity00) AS max_intensity_at_00

FROM minuteIntensitiesWide;

Result 6:

min_intensity_at_00	max_intensity_at_00
0	3

Explanation 6:

- The query calculates the minimum and maximum intensity values specifically for the first minute of each hour, showing the activity range at that timestamp (0-3 scale).

Query 7:

SELECT

COUNT(*) AS records_with_nulls

FROM minuteIntensitiesWide

WHERE

Id IS NULL OR

ActivityHour IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing values in critical columns (Id or ActivityHour).

Table 13: minuteMETsNarrow

Query 1:

```
PRAGMA table_info(minuteMETsNarrow);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityMinute	TEXT	0		0
2	METs	INTEGER	0		0

Explanation 1:

- The query retrieves schema metadata for the minuteMETsNarrow table.
- Helps understand the structure of minute-level MET values.

Query 2:

```
SELECT * FROM minuteMETsNarrow LIMIT 5;
```

Result 2:

Id	ActivityMinute	METs
1503960366	4/12/2016 12:00:00 AM	10
1503960366	4/12/2016 12:01:00 AM	10
1503960366	4/12/2016 12:02:00 AM	10
1503960366	4/12/2016 12:03:00 AM	10
1503960366	4/12/2016 12:04:00 AM	10

Explanation 2:

- The query retrieves the first 5 rows from the minuteMETsNarrow table.
- Helps verify data format and typical values.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM minuteMETsNarrow;
```

Result 3:

total_rows
1325580

Explanation 3:

- The query counts the total number of minute-level MET records in the minuteMETsNarrow table.
- Helps assess computational load for time-series analysis.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteMETsNarrow;
```

Result 4:

unique_participants
33

Explanation 4:

- The query counts the number of unique users (Id) in the minuteMETsNarrow table.
- Ensures data consistency with other minute-level tables.

Query 5:

```
SELECT
```

```
  MIN(ActivityMinute) AS earliest_datetime,
```

MAX(ActivityMinute) AS latest_datetime

FROM minuteMETsNarrow;

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:59:00 PM

Explanation 5:

- The query identifies the time range of the dataset by extracting the earliest (MIN) and latest (MAX) timestamps from the ActivityMinute column.
- Helps verify temporal alignment with other minute-level tables.

Query 6:

SELECT

MIN(METs) AS min_mets,

MAX(METs) AS max_mets

FROM minuteMETsNarrow;

Result 6:

min_mets	max_mets
0	157

Explanation 6:

- The query calculates the minimum and maximum MET values in the dataset, revealing the range of activity intensity.

Query 7:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM minuteMETsNarrow  
  
WHERE  
  
    Id IS NULL OR  
  
    ActivityMinute IS NULL OR  
  
    METs IS NULL;
```

Result 7:

records_with_nulls
0

Explanation 7:

- The query counts rows with missing values in critical columns, flagging potential data quality issues.

Table 14: minuteSleep

Query 1:

```
PRAGMA table_info(minuteSleep);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	date	TEXT	0		0
2	value	INTEGER	0		0
3	logId	INTEGER	0		0

Explanation 1:

- The query retrieves schema metadata for the minuteSleep table.
- Helps understand the structure of minute-level sleep data.

Query 2:

```
SELECT * FROM minuteSleep LIMIT 5;
```

Result 2:

Id	date	value	logId
1503960366	4/12/2016 2:47:30 AM	3	11380564589
1503960366	4/12/2016 2:48:30 AM	2	11380564589
1503960366	4/12/2016 2:49:30 AM	1	11380564589
1503960366	4/12/2016 2:50:30 AM	1	11380564589
1503960366	4/12/2016 2:51:30 AM	1	11380564589

Explanation 2:

- The query selects all data from the minuteSleep table.
- It limits the results to the first 5 rows.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM minuteSleep;
```

Result 3:

total_rows
188521

Explanation 3:

- Counts all rows in the minuteSleep table.
- Returns the total as total_rows.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteSleep;
```

Result 4:

unique_participants
24

Explanation 4:

- Counts distinct user IDs in the minuteSleep table.
- Returns the total as unique_participants.

Query 5:

SELECT

MIN(date) AS earliest_datetime,

MAX(date) AS latest_datetime

FROM minuteSleep;

Result 5:

earliest_datetime	latest_datetime
4/11/2016 10:00:00 PM	5/9/2016 9:59:30 PM

Explanation 5:

- Finds the earliest and latest dates in the minuteSleep table.
- Returns them as earliest_datetime and latest_datetime.

Query 6:

SELECT

MIN(value) AS min_sleep_value,

MAX(value) AS max_sleep_value

FROM minuteSleep;

Result 6:

min_sleep_value	max_sleep_value
1	3

Explanation 6:

- Returns the smallest and largest values in the value column.
- Labels results as min_sleep_value and max_sleep_value.

Query 7:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM minuteSleep  
  
WHERE  
  
    Id IS NULL OR  
  
    date IS NULL OR  
  
    value IS NULL;
```

Result 7:

records_with_nulls
0

Explanation 7:

- Counts records with NULL values in Id, date, or value columns.
- Returns total as records_with_nulls.

Table 15: minuteStepsNarrow

Query 1:

```
PRAGMA table_info(minuteStepsNarrow);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityMinute	TEXT	0		0
2	Steps	INTEGER	0		0

Explanation 1:

- Displays column structure of the minuteStepsNarrow table.
- Returns metadata like column names, types, and constraints.

Query 2:

```
SELECT * FROM minuteStepsNarrow LIMIT 5;
```

Result 2:

Id	ActivityMinute	Steps
1503960366	4/12/2016 12:00:00 AM	0
1503960366	4/12/2016 12:01:00 AM	0
1503960366	4/12/2016 12:02:00 AM	0
1503960366	4/12/2016 12:03:00 AM	0
1503960366	4/12/2016 12:04:00 AM	0

Explanation 2:

- Retrieves all columns from the minuteStepsNarrow table.
- Returns only the first 5 rows of data.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM minuteStepsNarrow;
```

Result 3:

total_rows
1325580

Explanation 3:

- Counts all records in the minuteStepsNarrow table.
- Returns the total as total_rows.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteStepsNarrow;
```

Result 4:

unique_participants
33

Explanation 4:

- Counts distinct user IDs in the minuteStepsNarrow table.
- Returns the total as unique_participants.

Query 5:

```
SELECT
```

```
    MIN(ActivityMinute) AS earliest_datetime,
```

```
    MAX(ActivityMinute) AS latest_datetime
```

FROM minuteStepsNarrow;

Result 5:

earliest_datetime	latest_datetime
4/12/2016 10:00:00 AM	5/9/2016 9:59:00 PM

Explanation 5:

- Finds the earliest and latest timestamps in the ActivityMinute column.
- Returns them as earliest_datetime and latest_datetime.

Query 6:

SELECT

MIN(Steps) AS min_steps,

MAX(Steps) AS max_steps

FROM minuteStepsNarrow;

Result 6:

min_steps	max_steps
0	220

Explanation 6:

- Returns the minimum and maximum values from the Steps column.
- Labels results as min_steps and max_steps.

Query 7:

SELECT

 COUNT(*) AS records_with_nulls

FROM minuteStepsNarrow

WHERE

 Id IS NULL OR

 ActivityMinute IS NULL OR

 Steps IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- Counts records with NULL values in any column.
- Returns total as records_with_nulls.

Table 16: minuteStepsWide

Query 1:

```
PRAGMA table_info(minuteStepsWide);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	ActivityHour	TEXT	0		0
2	Steps00	INTEGER	0		0
3	Steps01	INTEGER	0		0
4	Steps02	INTEGER	0		0
5	Steps03	INTEGER	0		0
6	Steps04	INTEGER	0		0
7	Steps05	INTEGER	0		0
8	Steps06	INTEGER	0		0
9	Steps07	INTEGER	0		0
10	Steps08	INTEGER	0		0
11	Steps09	INTEGER	0		0
12	Steps10	INTEGER	0		0
13	Steps11	INTEGER	0		0
14	Steps12	INTEGER	0		0
15	Steps13	INTEGER	0		0
16	Steps14	INTEGER	0		0
17	Steps15	INTEGER	0		0
18	Steps16	INTEGER	0		0
19	Steps17	INTEGER	0		0
20	Steps18	INTEGER	0		0
21	Steps19	INTEGER	0		0
22	Steps20	INTEGER	0		0
23	Steps21	INTEGER	0		0

24	Steps22	INTEGER	0		0
25	Steps23	INTEGER	0		0
26	Steps24	INTEGER	0		0
27	Steps25	INTEGER	0		0
28	Steps26	INTEGER	0		0
29	Steps27	INTEGER	0		0
30	Steps28	INTEGER	0		0
31	Steps29	INTEGER	0		0
32	Steps30	INTEGER	0		0
33	Steps31	INTEGER	0		0
34	Steps32	INTEGER	0		0
35	Steps33	INTEGER	0		0
36	Steps34	INTEGER	0		0
37	Steps35	INTEGER	0		0
38	Steps36	INTEGER	0		0
39	Steps37	INTEGER	0		0
40	Steps38	INTEGER	0		0
41	Steps39	INTEGER	0		0
42	Steps40	INTEGER	0		0
43	Steps41	INTEGER	0		0
44	Steps42	INTEGER	0		0
45	Steps43	INTEGER	0		0
46	Steps44	INTEGER	0		0
47	Steps45	INTEGER	0		0
48	Steps46	INTEGER	0		0
49	Steps47	INTEGER	0		0
50	Steps48	INTEGER	0		0
51	Steps49	INTEGER	0		0
52	Steps50	INTEGER	0		0
53	Steps51	INTEGER	0		0

[illegible]

[illegible]

Explanation 2:

- Retrieves all columns from the minuteStepsWide table.
- Returns only the first 5 rows of data

Query 3:

```
SELECT COUNT(*) AS total_rows FROM minuteStepsWide;
```

Result 3:

total_rows
21645

Explanation 3:

- Counts all records in the minuteStepsWide table.
- Returns the total as total_rows.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM minuteStepsWide;
```

Result 4:

unique_participants
33

Explanation 4:

- Counts distinct user IDs in the minuteStepsWide table.
- Returns the total as unique_participants.

Query 5:

```
SELECT  
  
    MIN(ActivityHour) AS earliest_datetime,  
  
    MAX(ActivityHour) AS latest_datetime  
  
FROM minuteStepsWide;
```

Result 5:

earliest_datetime	latest_datetime
4/13/2016 10:00:00 AM	5/9/2016 9:00:00 PM

Explanation 5:

- Finds the earliest and latest timestamps in the ActivityHour column.
- Returns them as earliest_datetime and latest_datetime.

Query 6:

```
SELECT  
  
    MIN(Steps00) AS min_steps_at_00,  
  
    MAX(Steps00) AS max_steps_at_00  
  
FROM minuteStepsWide;
```

Result 6:

min_steps_at_00	max_steps_at_00
0	186

Explanation 6:

- Returns the minimum and maximum values from the Steps00 column.
- Labels results as min_steps_at_00 and max_steps_at_00.

Query 7:

```
SELECT  
  
    COUNT(*) AS records_with_nulls  
  
FROM minuteStepsWide  
  
WHERE  
  
    Id IS NULL OR  
  
    ActivityHour IS NULL;
```

Result 7:

records_with_nulls
0

Explanation 7:

- Counts records with NULL values in Id or ActivityHour columns.
- Returns total as records_with_nulls.

Table 17: sleepDay

Query 1:

```
PRAGMA table_info(sleepDay);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	SleepDay	TEXT	0		0
2	TotalSleepRecords	INTEGER	0		0
3	TotalMinutesAsleep	INTEGER	0		0
4	TotalTimeInBed	INTEGER	0		0

Explanation 1:

- Displays column structure of the sleepDay table.
- Shows metadata including column names, data types, and constraints.

Query 2:

```
SELECT * FROM sleepDay LIMIT 5;
```

Result 2:

Id	SleepDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed
1503960366	4/12/2016 12:00:00 AM	1	327	346
1503960366	4/13/2016 12:00:00 AM	2	384	407
1503960366	4/15/2016 12:00:00 AM	1	412	442
1503960366	4/16/2016 12:00:00 AM	2	340	367
1503960366	4/17/2016 12:00:00 AM	1	700	712

Explanation 2:

- Retrieves all columns from the sleepDay table.
- Returns only the first 5 rows of data.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM sleepDay;
```

Result 3:

total_rows
413

Explanation 3:

- Counts all records in the sleepDay table.
- Returns the total as total_rows.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM sleepDay;
```

Result 4:

unique_participants
24

Explanation 4:

- Counts distinct user IDs in the sleepDay table.
- Returns the total as unique_participants.

Query 5:

```
SELECT  
  
    MIN(SleepDay) AS earliest_date,  
  
    MAX(SleepDay) AS latest_date  
  
FROM sleepDay;
```

Result 5:

earliest_date	latest_date
4/12/2016 12:00:00 AM	5/9/2016 12:00:00 AM

Explanation 5:

- Finds the earliest and latest dates in the SleepDay column.
- Returns them as earliest_date and latest_date.

Query 6:

```
SELECT  
  
    MIN(TotalSleepRecords) AS min_sleep_records,  
  
    MAX(TotalSleepRecords) AS max_sleep_records,  
  
    MIN(TotalMinutesAsleep) AS min_minutes_asleep,  
  
    MAX(TotalMinutesAsleep) AS max_minutes_asleep,  
  
    MIN(TotalTimeInBed) AS min_time_in_bed,  
  
    MAX(TotalTimeInBed) AS max_time_in_bed  
  
FROM sleepDay;
```

Result 6:

min_sleep_records	max_sleep_records	min_minutes_asleep	max_minutes_asleep	min_time_in_bed	max_time_in_bed
1	3	58	796	61	961

Explanation 6:

- Returns minimum and maximum values for all sleep metrics.
- Covers sleep records, minutes asleep, and time in bed.

Query 7:

SELECT

COUNT(*) AS records_with_nulls

FROM sleepDay

WHERE

Id IS NULL OR

SleepDay IS NULL OR

TotalMinutesAsleep IS NULL;

Result 7:

records_with_nulls
0

Explanation 7:

- Counts records with NULL values in key sleep tracking columns.
- Returns total as records_with_nulls.

Table 18: weightLogInfo

Query 1:

```
PRAGMA table_info(weightLogInfo);
```

Result 1:

cid	name	type	notnull	dflt_value	pk
0	Id	INTEGER	0		0
1	Date	TEXT	0		0
2	WeightKg	REAL	0		0
3	WeightPounds	REAL	0		0
4	Fat	TEXT	0		0
5	BMI	INTEGER	0		0
6	IsManualReport	TEXT	0		0
7	LogId	INTEGER	0		0

Explanation 1:

- Displays the schema structure of the weightLogInfo table.
- Shows column names, data types, and constraints.

Query 2:

```
SELECT * FROM weightLogInfo LIMIT 5;
```

Result 2:

Id	Date	WeightKg	WeightPounds	Fat	BMI	IsManualReport	LogId
1503960366	5/2/2016 11:59:59 PM	52.59999984741211	115.963146545323	22	22.64999996185303	True	1462233599000

1503960366	5/3/2016 11:59:59 PM	52.5999984741211	115.963146545323		22.6499996185303	True	1462319999000
1927972279	4/13/2016 1:08:52 AM	133.5	294.317120016975		47.5400009155273	False	1460509732000
2873212765	4/21/2016 11:59:59 PM	56.7000007629395	125.002104340889		21.4500007629395	True	1461283199000
2873212765	5/12/2016 11:59:59 PM	57.2999992370605	126.324874550011		21.6900005340576	True	1463097599000

Explanation 2:

- Retrieves all columns from the weightLogInfo table.
- Returns only the first 5 rows of data.

Query 3:

```
SELECT COUNT(*) AS total_rows FROM weightLogInfo;
```

Result 3:

total_rows
67

Explanation 3:

- Counts all records in the weightLogInfo table.
- Returns the total as total_rows.

Query 4:

```
SELECT COUNT(DISTINCT Id) AS unique_participants FROM weightLogInfo;
```

Result 4:

unique_participants
8

Explanation 4:

- Counts distinct user IDs in the weightLogInfo table.
- Returns the total as unique_participants.

Query 5:

```
SELECT  
  
    MIN(Date) AS earliest_datetime,  
  
    MAX(Date) AS latest_datetime  
  
FROM weightLogInfo;
```

Result 5:

earliest_datetime	latest_datetime
4/12/2016 11:59:59 PM	5/9/2016 6:39:44 AM

Explanation 5:

- Finds the earliest and latest timestamps in the Date column.
- Returns them as earliest_datetime and latest_datetime.

Query 6:

SELECT

```
MIN(WeightKg) AS min_weight_kg,  
MAX(WeightKg) AS max_weight_kg,  
MIN(BMI) AS min_bmi,  
MAX(BMI) AS max_bmi
```

FROM weightLogInfo;

Result 6:

min_weight_kg	max_weight_kg	min_bmi	max_bmi
52.5999984741211	133.5	21.4500007629395	47.5400009155273

Explanation 6:

- Returns minimum and maximum values for weight and BMI metrics.
- Covers both WeightKg and BMI columns.

Query 7:

SELECT

```
COUNT(*) AS records_with_nulls
```

FROM weightLogInfo

WHERE

```
Id IS NULL OR
```

```
Date IS NULL OR
```

```
WeightKg IS NULL;
```

Result 7:

records_with_nulls
0

Explanation 7:

- Counts records with NULL values in key weight tracking columns.
- Returns total as records_with_nulls.