```
--fitbit_sql_analysis.sql

-- run as postgres superuser
CREATE DATABASE fitbit;
CREATE USER fitbit_user WITH PASSWORD 'YourStrongPassword';
GRANT ALL PRIVILEGES ON DATABASE fitbit TO fitbit_user;

-- 1) create a raw table
CREATE TABLE raw_daily_activity (
  id TEXT,
  activitydate TEXT,
  totalsteps TEXT,
  totaldistance TEXT,
  calories TEXT,
  veryactiveminutes TEXT,
  fairlyactiveminutes TEXT,
  lightlyactiveminutes TEXT,
  sedentaryminutes TEXT
);

-- 2) from psql terminal use \copy (local file path)

-- 3) Quick inspection (sanity checks)
-- total rows
SELECT COUNT(*) FROM raw_daily_activity;

-- first 5 rows
SELECT * FROM raw_daily_activity LIMIT 5;

-- check how many NULL/empty in important columns
SELECT
  COUNT(*) AS total_rows,
  COUNT(NULLIF(TRIM(totalsteps), '')) AS present_totalsteps,
  COUNT(NULLIF(TRIM(calories), '')) AS present_calories
FROM raw_daily_activity;

-- trim whitespace (optional)
UPDATE raw_daily_activity
SET totalsteps = TRIM(totalsteps),
    calories = TRIM(calories),
    activitydate = TRIM(activitydate),
    totaldistance = TRIM(totaldistance);


-- 5) Basic cleaning (trim whitespace, turn empty strings to NULL)
-- replace empty strings with NULL for columns used
UPDATE raw_daily_activity
SET totalsteps = NULLIF(totalsteps, ''),
    calories = NULLIF(calories, ''),
    totaldistance = NULLIF(totaldistance, ''),
    activitydate = NULLIF(activitydate, '');


-- 6) Create a typed cleaned table (convert strings → proper types)
CREATE TABLE cleaned_daily_activity AS
SELECT
  id,
```

```
  -- if activitydate like 4/12/2016 use 'MM/DD/YYYY', else use
activitydate::date if 'YYYY-MM-DD'
  CASE
    WHEN activitydate ~ '^[0-9]{1,2}/[0-9]{1,2}/[0-9]{4}$' THEN
TO_DATE(activitydate, 'MM/DD/YYYY')
    ELSE NULLIF(activitydate, '')::DATE
  END AS activity_date,
  NULLIF(totalsteps,'')::INTEGER AS total_steps,
  NULLIF(totaldistance,'')::NUMERIC AS total_distance,
  NULLIF(calories,'')::INTEGER AS calories,
  NULLIF(veryactiveminutes,'')::INTEGER AS very_active_minutes,
  NULLIF(fairlyactiveminutes,'')::INTEGER AS fairly_active_minutes,
  NULLIF(lightlyactiveminutes,'')::INTEGER AS lightly_active_minutes,
  NULLIF(sedentaryminutes,'')::INTEGER AS sedentary_minutes
FROM raw_daily_activity;

-- Check cleaned_daily_activity:
SELECT COUNT(*) FROM cleaned_daily_activity;
SELECT * FROM cleaned_daily_activity LIMIT 5;

-- 7) Remove duplicates (if same Id + date repeats)
CREATE TABLE deduped_daily_activity AS
SELECT id, activity_date, total_steps, total_distance, calories,
       very_active_minutes, fairly_active_minutes,
lightly_active_minutes, sedentary_minutes
FROM (
  SELECT *,
         ROW_NUMBER() OVER (PARTITION BY id, activity_date ORDER BY
total_steps DESC NULLS LAST) AS rn
  FROM cleaned_daily_activity
) t
WHERE rn = 1;

--Verify:
SELECT COUNT(*) FROM deduped_daily_activity;
SELECT COUNT(*) FROM (
  SELECT id, activity_date, COUNT(*) FROM deduped_daily_activity GROUP
BY id, activity_date HAVING COUNT(*) > 1
) x;

--8) Add indexes for faster queries (recommended)
CREATE INDEX idx_dedup_id ON deduped_daily_activity (id);
CREATE INDEX idx_dedup_date ON deduped_daily_activity (activity_date);

--9) Useful analysis queries to run & save (these go into your SQL
deliverable)
-- A. Basic stats
SELECT COUNT(DISTINCT id) AS total_users, COUNT(*) AS total_records
FROM deduped_daily_activity;

-- B. Average steps per day (overall)
SELECT ROUND(AVG(total_steps)::numeric,2) AS avg_steps_overall FROM
deduped_daily_activity;

-- C. Average steps per user (top 20)
SELECT id, ROUND(AVG(total_steps)::numeric,2) AS avg_steps
FROM deduped_daily_activity
GROUP BY id
```

```
ORDER BY avg_steps DESC
LIMIT 20;

-- D. Correlation between steps and calories (Postgres has corr)
SELECT corr(total_steps::double precision, calories::double precision)
AS steps_calories_corr
FROM deduped_daily_activity;

-- E. Avg active minutes breakdown
SELECT
  ROUND(AVG(very_active_minutes)::numeric,2) AS avg_very,
  ROUND(AVG(fairly_active_minutes)::numeric,2) AS avg_fairly,
  ROUND(AVG(lightly_active_minutes)::numeric,2) AS avg_light,
  ROUND(AVG(sedentary_minutes)::numeric,2) AS avg_sedentary
FROM deduped_daily_activity;

-- F. Steps by day of week
SELECT EXTRACT(DOW FROM activity_date) AS dow,
       ROUND(AVG(total_steps)::numeric,2) AS avg_steps
FROM deduped_daily_activity
GROUP BY dow
ORDER BY dow;

-- create sleep table 1st {Raw Sleep CSV import karo}
CREATE TABLE raw_sleep_day (
  id TEXT,
  sleepdate TEXT,
  totalminutesasleep TEXT,
  totaltimeinbed TEXT
);
-- import csv file directly
--Clean table banao (datatype convert)
CREATE TABLE cleaned_sleep_day AS
SELECT
  id,
  TO_DATE(sleepdate, 'MM/DD/YYYY') AS sleep_date,
  NULLIF(totalminutesasleep,'')::INTEGER AS total_minutes_asleep,
  NULLIF(totaltimeinbed,'')::INTEGER AS total_time_in_bed
FROM raw_sleep_day;

--Deduplicate (agar ek user+date multiple rows hai)
CREATE TABLE deduped_sleep_day AS
SELECT id, sleep_date, total_minutes_asleep, total_time_in_bed
FROM (
  SELECT *,
         ROW_NUMBER() OVER (PARTITION BY id, sleep_date ORDER BY
total_minutes_asleep DESC NULLS LAST) AS rn
  FROM cleaned_sleep_day
) t
WHERE rn = 1;

--Check data
SELECT COUNT(*) FROM deduped_sleep_day;
SELECT * FROM deduped_sleep_day LIMIT 10;


-- G. Join with sleep table (if you have sleep csv)
```

```
-- assume you loaded/cleaned 'deduped_sleep_day' with column sleep_date
(date) and total_minutes_asleep
SELECT d.id, d.activity_date, d.total_steps, s.total_minutes_asleep
FROM deduped_daily_activity d
LEFT JOIN deduped_sleep_day s
  ON d.id = s.id AND d.activity_date = s.sleep_date
LIMIT 50;

-- 10) Create views to make reporting easier
CREATE VIEW v_user_daily_summary AS
SELECT id, activity_date,
       total_steps, total_distance, calories,
       very_active_minutes, lightly_active_minutes, sedentary_minutes
FROM deduped_daily_activity;

-- now we can see by running query
SELECT * FROM v_user_daily_summary WHERE id='1503960366';

-- 11) Export query results for PDF/report
--save it directly from data (save as excel or pdf)

-- 1) Raw sleep table (if not exists)
CREATE TABLE IF NOT EXISTS raw_sleep_day (
  id TEXT,
  sleepday TEXT,
  totalsleeprecords TEXT,
  totalminutesasleep TEXT,
  totaltimeinbed TEXT
);

-- 2) Cleaned table: parse datetime to date
CREATE TABLE IF NOT EXISTS cleaned_sleep_day AS
SELECT
  id,
  TO_TIMESTAMP(sleepdate, 'MM/DD/YYYY hh12:mi:ssAM')::date AS
sleep_date,
  NULLIF(totalminutesasleep,'')::int AS total_minutes_asleep,
  NULLIF(totaltimeinbed,'')::int AS total_time_in_bed
FROM raw_sleep_day;

-- 3) Deduped table (keep max sleep minutes per Id+date)
CREATE TABLE IF NOT EXISTS deduped_sleep_day AS
SELECT id, sleep_date, total_minutes_asleep, total_time_in_bed
FROM (
  SELECT *,
         ROW_NUMBER() OVER (PARTITION BY id, sleep_date ORDER BY
total_minutes_asleep DESC NULLS LAST) AS rn
  FROM cleaned_sleep_day
) t
WHERE rn = 1;


--validation
SELECT COUNT(*) FROM deduped_sleep_day;
SELECT * FROM deduped_sleep_day LIMIT 10;
SELECT id, sleep_date, total_minutes_asleep
FROM deduped_sleep_day
ORDER BY sleep_date DESC
```

```
LIMIT 5;
```