1. Data Ingestion & Orchestration

In[698]:=
```
(*Automation through terminal and shell script to move CSV file to target folder*)
```

In[699]:=
```
terminal[string_String] := RunProcess[{"zsh", "-i", "-c", string}];
Hold[terminal["data_ingest.sh"]];
```

In[701]:=
```
(*

Note on Live Execution:
  To run the ingestion automation live,ensure:
 1.The data_ingest.sh script is in the
  terminal's default working directory (typically $HOME).
   2.Execution permissions are set (chmod+x data_ingest.sh).
   3.Valid CSV in ~/Downloads
  4.The Hold[] wrapper is removed from the command above.

   Demo Mode (Default):If dependencies are missing,
the system automatically falls back to embedded Sample Data
 to demonstrate the visualization logic without configuration.

*)
```

2. Data Transformation & Validation

Data & Sample

In[702]:=
```
(*Define the live path*)
livePath = StringJoin[$HomeDirectory, "/Documents/HealthData/raw_data.csv"]
```

Out[702]=
```
/Users/andrefreitas/Documents/HealthData/raw_data.csv
```

In[703]:=
```
(*Import the data*)
```

In[704]:=
```
(*The Logic:Try to find the file,
otherwise load sample to run in DEMO mode*)data = If[FileExistsQ[livePath],
   Import[livePath, "RawData"], (*ELSE:Print a message and use sample*)
   Print["   Live file not found. Switching to DEMO MODE with sample data."];
   {···} + ];
```

```
Live file not found. Switching to DEMO MODE with sample data.
```

In[705]:=
```
(*Small sample of the data filtered by type "sleep"*)
```

In[706]:=
```
{First[data], Select[data, Function[#[[1]] === "Sleep"]][[1 ;; 3]]} // TableForm;
```

Variables1

In[707]:=
```
(*Putting data into Tabular format,
filtering for Sleep and selecting only the relevant columns (first 4)*)
```

In[708]:=
```
sleepTabular1 =
 Tabular[Select[data, Function[#[[1]] === "Sleep"]], First[data]][[All, 1 ;; 4]]
```

Out[708]=

	Type	Start	End	Duration
1	Sleep	2026–01–30 18:35	2026–01–31 07:27	12:52
2	Sleep	2026–01–30 14:23	2026–01–30 15:07	00:44
3	Sleep	2026–01–30 10:11	2026–01–30 10:52	00:41
4	Sleep	2026–01–30 04:15	2026–01–30 07:28	03:13
5	Sleep	2026–01–29 23:40	2026–01–30 04:01	04:21
6	Sleep	2026–01–29 18:40	2026–01–29 23:29	04:49
7	Sleep	2026–01–29 13:12	2026–01–29 15:03	01:51

In[709]:=
```
(*adding a column 'Date' to indicate the day in which the sleeping started*)
```

In[710]:=
```
sleepTabular2 = TransformColumns[sleepTabular1,
  {"Type", "Date" → Function[StringSplit[#Start, " "] /. {a_, b_} → a]}]
```

Out[710]=

	Type	Date	Start	End	Duration
1	Sleep	2026–01–30	2026–01–30 18:35	2026–01–31 07:27	12:52
2	Sleep	2026–01–30	2026–01–30 14:23	2026–01–30 15:07	00:44
3	Sleep	2026–01–30	2026–01–30 10:11	2026–01–30 10:52	00:41
4	Sleep	2026–01–30	2026–01–30 04:15	2026–01–30 07:28	03:13
5	Sleep	2026–01–29	2026–01–29 23:40	2026–01–30 04:01	04:21
6	Sleep	2026–01–29	2026–01–29 18:40	2026–01–29 23:29	04:49
7	Sleep	2026–01–29	2026–01–29 13:12	2026–01–29 15:03	01:51

In[711]:=
```
(*Variable that list all days in string ISODate format from when sleeping
 data started to be collected (17/05/2025) up to the current day*)
```

In[712]:=
```
dates =
  (DateString[#, "ISODate"] & /@ DateRange[DateObject["2025-05-17"], Today, "Day"]);
```

Functions

In[713]:=
```
(*Function that adds column "Duration2" and "Time Range" which gives
 the duration of each sleeping session restricted to a specific day*)
```

In[714]:=
```
f[date_String] := TransformColumns[Select[sleepTabular2,
    Function[(StringSplit[#End, " "] /. {a_, b_} → a) == date || #Date == date]],
  {"Duration2" → Function[If[(StringSplit[#End, " "] /. {a_, b_} → a) ≠ date,
      DateObject[StringJoin[date, " 23:59"]] - DateObject[#Start],
      If[(StringSplit[#Start, " "] /. {a_, b_} → a) ≠ date,
       DateObject[#End] - DateObject[StringJoin[date, " 00:01"]], DateObject[#End] -
        DateObject[#Start]], DateObject[#End] - DateObject[#Start]]],
   "TimeRange" → Function[If[(StringSplit[#End, " "] /. {a_, b_} → a) ≠ date,
      (StringSplit[#Start, " "] /. {a_, b_} → b) <> "-23:59",
      If[(StringSplit[#Start, " "] /. {a_, b_} → a) ≠ date,
       "00:00-" <> (StringSplit[#End, " "] /. {a_, b_} → b),
       (StringSplit[#Start, " "] /. {a_, b_} → b) <> "-" <>
        (StringSplit[#End, " "] /. {a_, b_} → b)]]]}]
```

In[715]:=
```
(*example: the column "duration2" displays the amount of sleep only up until 23:
 59 on the 21/05/2025*)
```

In[716]:=
```
f["2025-05-21"]
```

Out[716]=

	Type	Date	Start	End	Duration	Duration2 (min)	TimeRa
1	Sleep	2025–05–21	2025–05–21 23:45	2025–05–22 00:10	00:24	14	23:45–
2	Sleep	2025–05–21	2025–05–21 19:54	2025–05–21 20:50	00:56	56	19:54–
3	Sleep	2025–05–21	2025–05–21 18:47	2025–05–21 19:00	00:13	13	18:47–
4	Sleep	2025–05–21	2025–05–21 16:35	2025–05–21 18:46	02:11	131	16:35–
5	Sleep	2025–05–21	2025–05–21 15:15	2025–05–21 15:27	00:12	12	15:15–
6	Sleep	2025–05–21	2025–05–21 13:00	2025–05–21 14:35	01:35	95	13:00–

In[717]:=
```
(*testing I can use the function to calculate total amount of sleep from 00:
 00 to 23:59 on any given day*)
```

In[718]:=
```
f["2025-05-21"][[All, "Duration2"]] // Normal // Total
```

Out[718]=
```
932 min
```

In[719]:=
```
(*Function Gives the total amount of sleep for the inputed day,
includint partial days*)
```

In[720]:=
```
totalDaySleep[date_String] :=
 (f[date][[All, 6]] // Normal // Total) /. Quantity[x_, "Minutes"] →
   Quantity[MixedMagnitude[{0, x}], MixedUnit[{"Hours", "Minutes"}]]
```

In[721]:=
```
(*Example*)
```

In[722]:=
```
totalDaySleep["2026-01-25"]
```

Out[722]=
```
14h 33min
```

In[723]:=
```
(*Function gives last x days (including current day) as string "ISODate"*)
```

In[724]:=
```
listOfDates[daysback_Integer] := (DateString[#, "ISODate"] & /@
   DateRange[DatePlus[Today, -daysback], Yesterday, "Day"])
```

In[725]:=
```
(*Example*)
```

In[726]:=
```
listOfDates[7]
```

Out[726]=
```
{2026-01-24, 2026-01-25, 2026-01-26, 2026-01-27, 2026-01-28, 2026-01-29, 2026-01-30}
```

In[727]:=
```
(*Function Gives the total amount of sleep per
 day in the last "daysback"s not including current day*)
```

In[728]:=
```
totalDaySleep2[daysback_Integer] := Rule @@@ Partition[
   Riffle[listOfDates[daysback], Map[totalDaySleep[#] &, listOfDates[daysback]]], 2]
```

In[729]:=
```
(*Example*)
```

In[730]:=
```
totalDaySleep2[3]
```

Out[730]=
```
{2026-01-28 → 14h 36min, 2026-01-29 → 13h 27min, 2026-01-30 → 14h 2min}
```

Variables 2

Defining secondary feature set based on the primary variables established in Variables 1 subsection and on the functions established in the Functions section.

In[731]:=
```
(*All data on total sleep (in hours and minutes) per day as an ASSOCIATION*)
```

In[732]:=
```
sleepData = Rule @@@ Partition[Riffle[dates, Map[totalDaySleep, dates]], 2];
```

In[733]:=
```
(*All data on total sleep per day as list of decimal hours*)
```

In[734]:=
```
sleepData2 = Drop[Function[UnitConvert[#, "Hours"]] /@ Values[sleepData], -1] // N;
```

Insights & Visual Monitoring

Total Hours per day

In[735]:=
```
(*Function to manually calculate the total amount of sleep in a day,
for partial days calculations*)
```

In[736]:=
```
manualSleeping[hours_List, minutes_List] :=
 Quantity[MixedMagnitude[{Total[Join[hours, {IntegerPart[Total[minutes] / 60]}]],
    FractionalPart[Total[minutes] / 60] * 60}], MixedUnit[{"Hours", "Minutes"}]]
```

In[737]:=
```
(*Example*)
```

In[738]:=
```
(*2026-01-31*)
```

In[739]:=
```
manualSleeping[{1, 4}, {23, 34}]
```

Out[739]=
```
5h 57min
```

In[740]:=
```
(*Total sleep in current day - Almost always partial*)
```

In[741]:=
```
totalDaySleep[DateString[Today, "ISODate"]]
```

Out[741]=
```
7h 26min
```

In[742]:=
```
(*Total sleep per day from the last 8 days*)
```

In[743]:=
```
totalDaySleep2[7]
```

Out[743]=
```
{2026-01-24 → 14h 6min, 2026-01-25 → 14h 33min,
 2026-01-26 → 13h 46min, 2026-01-27 → 14h 0min,
 2026-01-28 → 14h 36min, 2026-01-29 → 13h 27min, 2026-01-30 → 14h 2min}
```

In[744]:=
```
(*Avarege sleep from the last 7 days*)
```

In[745]:=
```
((Mean[
      UnitConvert[#, "Minutes"] & /@
       (743[[All, 2]])]
     // N)
   / 60.0) /. Quantity[a_, "Minutes"] :>
  Quantity[MixedMagnitude[{IntegerPart[a], IntegerPart[FractionalPart[a] * 60]}],
   MixedUnit[{"Hours", "Minutes"}]]
```

Out[745]=
```
14h 4min
```

Naps per day

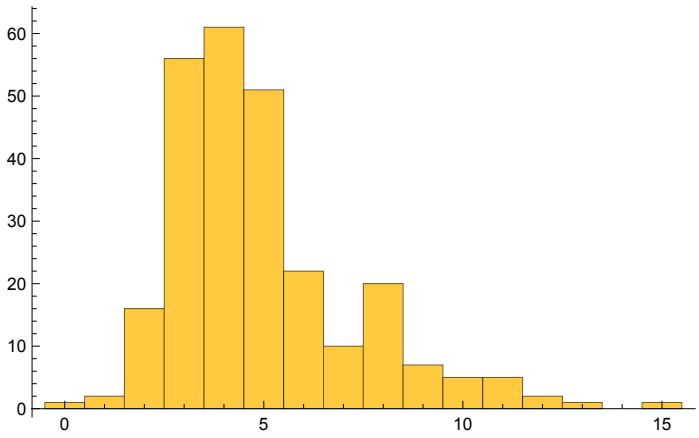
In[746]:=
```
(*tracking the number of naps (under 120 min) per day*)
```

In[747]:=
```
Function[Length[Select[f[#], Function[#Duration2 < 120]]]] /@ dates
```

Out[747]=
```
{8, 8, 15, 11, 8, 9, 5, 8, 10, 10, 7, 8, 9, 10, 11, 9, 6, 11, 11, 12, 9, 8, 8, 13, 8, 10, 12,
 8, 8, 8, 9, 8, 7, 8, 5, 9, 9, 6, 6, 11, 8, 7, 8, 8, 7, 4, 6, 10, 6, 5, 8, 5, 5, 6, 6, 5,
 6, 6, 6, 5, 6, 4, 6, 5, 4, 6, 5, 2, 5, 5, 5, 3, 5, 5, 6, 5, 7, 2, 8, 7, 4, 2, 4, 4, 4, 5,
 4, 4, 3, 6, 3, 6, 3, 4, 5, 4, 5, 6, 4, 6, 3, 3, 4, 3, 3, 4, 5, 4, 6, 4, 3, 4, 4, 7, 5,
 4, 6, 5, 3, 5, 5, 5, 5, 5, 7, 3, 5, 3, 5, 5, 4, 3, 1, 4, 5, 5, 5, 4, 5, 3, 4, 4, 3, 4,
 4, 4, 3, 3, 3, 5, 6, 8, 4, 4, 3, 4, 4, 7, 3, 4, 3, 5, 4, 4, 4, 3, 4, 2, 3, 5, 3, 4, 5,
 3, 3, 5, 5, 4, 3, 3, 4, 4, 4, 4, 3, 5, 5, 4, 3, 3, 4, 3, 4, 4, 5, 4, 3, 5, 5, 3, 7, 8,
 5, 5, 3, 4, 3, 4, 5, 4, 3, 5, 3, 4, 3, 6, 3, 4, 4, 3, 3, 4, 3, 3, 3, 5, 4, 4, 4, 5, 5,
 1, 3, 2, 3, 3, 2, 3, 3, 4, 3, 3, 3, 2, 3, 3, 3, 2, 4, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 0}
```

In[748]:=
```
Histogram[747]
```

Out[748]=



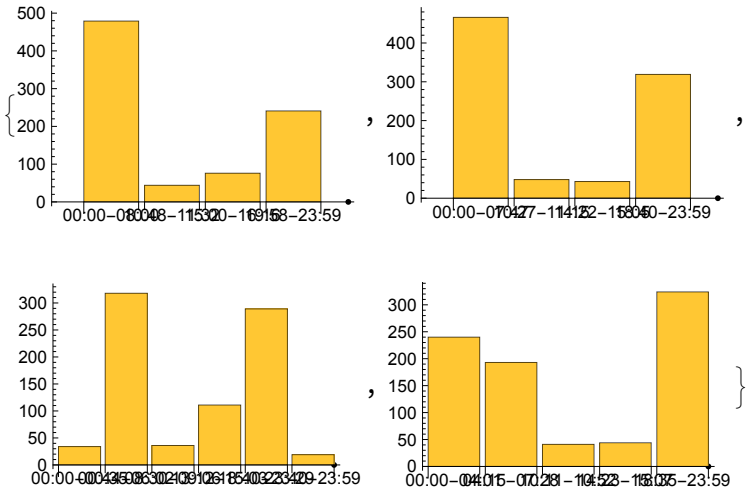
Distribution of Naps along specific days

In[749]:=
```
barChartSleep[date_String] :=
 BarChart[f[date][[All, "Duration2"]] // QuantityMagnitude // Normal // Reverse,
  ChartLabels → (f[date][[All, "TimeRange"]] // Normal // Reverse)]
```

In[750]:=

```
barChartSleep[#] & /@ listOfDates[4]
```

Out[750]=



Moving average total hours per day

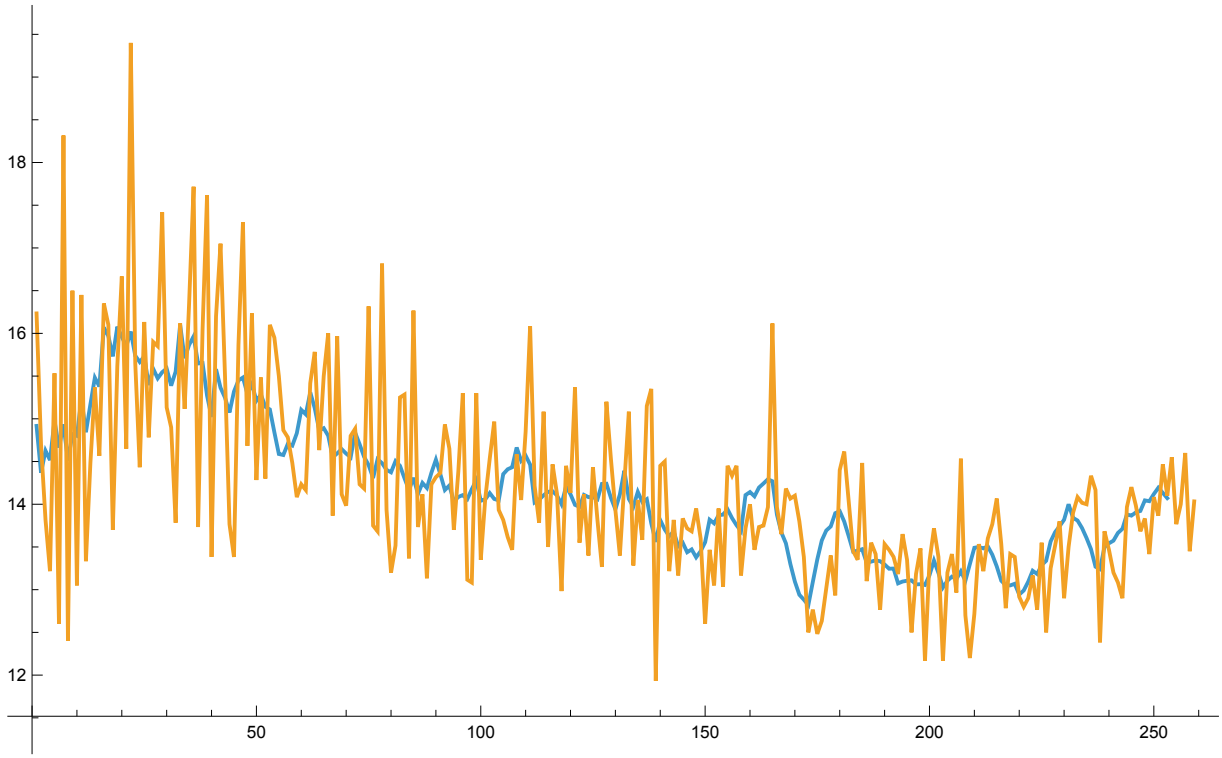
In[751]:=

```
(*Moving average for 7 days agains raw data*)
```

In[752]:=

```
ListLinePlot[{MovingAverage[sleepData2, 7], sleepData2}]
```

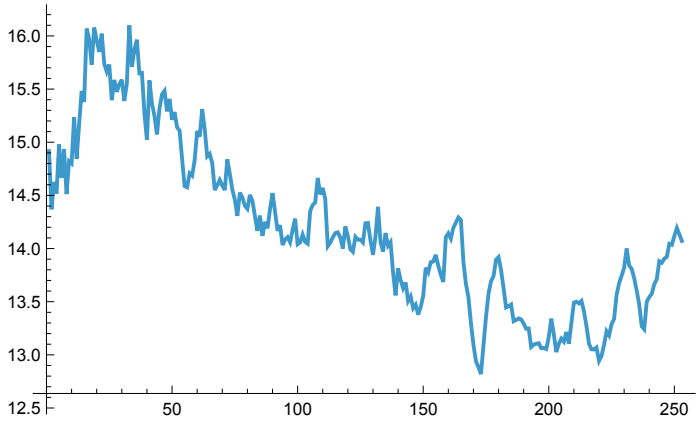
Out[752]=



In[753]:=
```
(*Moving average only*)
```

In[754]:=
```
ListLinePlot[MovingAverage[sleepData2, 7]]
```

Out[754]=



In[755]:=
```
(*Fitting the data with ListFitPlot[]*)
```

In[756]:=
```
ListFitPlot[sleepData2[[1 ;; -2]], PlotRange → {Automatic, {12, 20}}]
```

Out[756]=