Vehicle Event & Charge-Analytics — Take-Home Brief

We care more about **how you think** than about following an instruction sheet.
Everything below defines *what* must be true at the end—not *how* to get there.

1 Data Resources (discover the schemas)

Tag	Feed	Typical role
TLM	High-rate telematics snapshots	ignition flag, battery %, speed, odometer…
TRG	Low-rate trigger log	multiple `name/val` pairs for each timestamp
MAP	Vehicle ↔ PNID map	links primary vehicle IDs (TLM) to all PNIDs (TRG)
SYN	Synthetic overrides	curated "ignition-off" moments by vehicle

All files are CSV except **SYN** (JSON). Column names, dtypes, and edge-cases are intentionally *not* documented—explore and state your assumptions.

2 Domain Vocabulary

Term	Description
Ignition Status	Engine / powertrain on ↔ off.
Ignition Event	First time a status *change* is observed (via TLM, TRG `IGN_CYCL` , or SYN override).
Battery Snapshot	Any battery-percentage reading from TLM or from TRG where `name = CHARGE_STATE` .
Charging Event	A meaningful battery-% rise over time (you define thresholds; see §4).

3 Your Tasks

1. **Data sanity pass**

   Verify that timestamps parse cleanly, units make sense, ranges are plausible, and joins line up.

   - Flag / fix malformed records, obvious outliers, or clock drift.
   - Document every anomaly you detect and how you handled (or would handle) it.

2. **Ignition-event extraction**

   Produce **IgnitionEvents** with exactly:
   `vehicle_id | event | event_ts`
   event ∈ { `ignitionOn` , `ignitionOff` }.

   Extract from 3 sources:

   i. TLM
   ii. TRG

iii. SYN

3. **Charging Status Events Extraction**

   Produce **ChargingStatusEvents** with exactly:
   `vehicle_id | event | event_ts`
   event ∈ { `Active` , `Abort` , `Completed` }.
   Extract from TRG

4. **Battery Level Association**

   Find the closest battery reading within **±300 s** of any candidate event (Ignition Event or Charging Status Event).
   If two readings tie, defend your tie-breaker.
   If no reading is that close, treat the charge level as unknown.

You may use **Python, SQL, or a mix**—choose what lets you reason clearly.

4 Charging-event detection

Using EV battery level changes at Candidate events, produce **ChargingEvents**. Schema is up to you, but include:

- vehicle_id
- start_ts / end_ts (or single ts)
- Δ battery %
- ignition_state at detection
  Explain and justify:
- the %-jump that qualifies as “real” (hint: stricter when the engine is ON),
- how you debounce noise and avoid double-counting within one continuous charge session.

5 Evaluation Criteria

Weight	What we look for
40 %	Correctness & robustness of event logic (edge-cases, multi-source fusion)
25 %	Code clarity, structure, and reproducibility
20 %	Thoroughness of your data-quality review
15 %	Clear write-up of assumptions, trade-offs, and next-steps

6 Submission Checklist

1. **repo / zip** with code (Python and/or SQL) and a one-command run script
2. Two CSV/Parquet outputs: **IgnitionEvents**, **ChargingEvents**
3. **README** covering
   - discovered schemas & data issues,
   - design choices,
   - “what I’d improve next” (≤ 300 words)

That’s it—show us your reasoning!