Build LP Panel — Direct-From-Raw Strategy & Stepwise Debug Plan

This report documents **exactly how** we will construct the six LP time series directly from the three raw data sources (no intermediate “normalization” layer). It also lays out a **stepwise build-and-debug plan** so we only proceed to the next phase when the current one passes deterministic QA gates.

---

1) Objective

Construct six LP series that respect the pre/post-reform data grain while keeping the economic definition **LP = w × Π** intact:

1. **Haifa (port)** — **Monthly**: 2018-01 → 2021-08
2. **Ashdod (port)** — **Monthly**: 2018-01 → 2021-08
3. **Haifa-Legacy (terminal)** — **Quarterly**: 2021-Q3 → 2024-Q4
4. **Haifa-SIPG/Bayport (terminal)** — **Quarterly**: 2021-Q3 → 2024-Q4
5. **Ashdod-Legacy (terminal)** — **Quarterly**: 2021-Q3 → 2024-Q4
6. **Ashdod-HCT/ACH/TIL (terminal)** — **Quarterly**: 2021-Q3 → 2024-Q4

   Ignore all rows for **Eilat** and **All Ports** everywhere.

---

2) Inputs (no normalization layer)

A) Monthly tons (ports & terminals)

Path: `Data/Output/monthly_output_by_1000_tons_ports_and_terminals.tsv`
Columns: `PortOrTerminal`, `Month-Year`, `tons_k`

- `PortOrTerminal ∈ {Ashdod, Haifa}` → **port-level** monthly tons.
- `PortOrTerminal ∈ {Ashdod HCT, Haifa SIPG}` → **terminal-level** monthly tons.
- `PortOrTerminal = All Ports` → drop.
- **Units:** tons = `tons_k × 1000`.
- **Port-month precedence rule:** if *any* terminal rows exist for `(port,year,month)`, define `tons_port_m = Σ_terminal tons_i_m`; otherwise `tons_port_m = tons_portrow_m` (the single port row).

B) TEU mixed frequency (monthly & quarterly in one file)

Path: `Data/Output/teu_monthly_plus_quarterly_by_port.tsv`
Columns: `Port`, `Period`, `Freq`, `Year`, `MonthIndex`, `TEU_thousands`, `TEU`

- Use `TEU` when present; otherwise `TEU := TEU_thousands × 1000`.
- **Monthly TEU (pre-reform ports):** `Freq = 'Monthly'` & `Port ∈ {Ashdod, Haifa}` → `(port,year,month, TEU_port_m)`.
- **Quarterly TEU (post-reform terminals):** `Freq = 'Quarterly'` & `Port ∈ {Haifa, Haifa SIPG, Ashdod, Ashdod HCT}` → map to terminal names (see below) and parse `(port, terminal, year, quarter, TEU_i_q)`.
- **Port-quarter TEU:** `TEU_port_q = Σ_terminal TEU_i_q` within `(port,year,quarter)`.

C) Terminal×month Labor & Π

Path: `Data/L_proxy/L_Proxy.tsv`
Key columns: `port`, `terminal`, `year`, `month`, `quarter`, `L_hours_i_m`, `Pi_teu_per_hour_i_y`, plus `TEU_i_m`, `share_i_p_q` for mix computation.

- Grain is **terminal × month**.
- **Π_{iy}** (`Pi_teu_per_hour_i_y`) is terminal-year intrinsic productivity.
- **Port labor** at month = `Σ_i L_hours_i_m` across terminals.

Terminal name mapping (ensure consistent joins)

- TEU `Port='Haifa SIPG'` → L_Proxy `terminal='Haifa-Bayport'`
- TEU `Port='Ashdod HCT'` → L_Proxy `terminal='Ashdod-HCT'`
- TEU `Port='Haifa'` (Quarterly) → L_Proxy `terminal='Haifa-Legacy'`
- TEU `Port='Ashdod'` (Quarterly) → L_Proxy `terminal='Ashdod-Legacy'`

---

3) LP math (unchanged)

3.1 Monthly port LP (pre-reform)

- **Ratio:** `r_{p,m} = tons_{p,m} / TEU_{p,m}`.
- **Winsorize & Rebase by (port,year):** winsorize `r` to `[1%, 99%]`; rebase to mean 1 ⇒ `w_{p,m}`.
- **Mix base Π (month):** quarter-constant terminal shares (`shares_{i,p,q}` from `TEU_i_m` or `share_i_p_q`) times Π_{iy}, broadcast to months ⇒ `Π_{p,m}`.
- **LP:** `LP_{p,m} = w_{p,m} × Π_{p,m}`.
- **Diagnostic identity:** `LP_id = TEU_{p,m} / Σ_i L_hours_{i,m}` when both positive.

3.2 Quarterly terminal LP (post-reform)

- **Port ratio per quarter:** `r_{p,q} = (Σ_{m∈q} tons_{p,m}) / TEU_{p,q}` where `TEU_{p,q} = Σ_i TEU_{i,q}`.

- **Winsorize & Rebase by (port,year):** ⇒ `w_{p,q}`.
- **Terminal LP:** `LP_{i,q} = w_{p(i),q} × Π_iy`.

---

4) Output artifacts (six atomic series + optional long panel)

1) `LP_Haifa_port_month.tsv` (2018-01..2021-08)
2) `LP_Ashdod_port_month.tsv` (2018-01..2021-08)
3) `LP_Haifa_Legacy_quarter.tsv` (2021-Q3..2024-Q4)
4) `LP_Haifa_SIPG_quarter.tsv` (2021-Q3..2024-Q4)
5) `LP_Ashdod_Legacy_quarter.tsv` (2021-Q3..2024-Q4)
6) `LP_Ashdod_HCT_quarter.tsv` (2021-Q3..2024-Q4)

Each row includes: `port`, `terminal` (if any), `year`, `month` (if monthly), `month_index` (if monthly), `quarter` (if quarterly), `TEU`, `tons`, `w`, `w_source ∈ {monthly, quarterly}`, `Pi`, `LP`, and provenance (`tons_source`).

Optional convenience: `LP_panel_long.tsv` stacking the six with `series_id` and `freq ∈ {M,Q}`.

---

5) Stepwise Build-and-Debug Plan (gated progression)

We move in **four gated stages**. Each stage writes its own outputs and a QA report. We proceed **only if the QA passes**. If a stage fails, we fix inputs/logic and re-run that stage; later stages never run on bad prerequisites.

Stage 0 — Configuration & Guards

- Set constants: cutover month (`2021-09` → monthly ends at `2021-08`), ranges, winsor bounds (`1%, 99%`).
- Define terminal mapping dictionary.
- **Guardrails activated:** drop Eilat & All Ports; enforce month∈[1..12], quarter∈{Q1..Q4}.
- **Deliverables:** `_meta_config.json` (params), `_stage0_ok` marker.

QA-0 (must-pass): - Terminal mapping covers all terminals seen in TEU quarterly and L_Proxy.
- No unexpected ports/terminals after filtering.

Stage 1 — Build tons tables (from monthly tons file only)

- Parse `Month-Year` → `(year, month)`; `month_index = year*12 + month`.
- Build **terminal-month tons** for `{Ashdod-HCT, Haifa-SIPG}`.
- Build **port-month tons** with precedence: `sum terminals` if present else `port row`.
- Aggregate **port-quarter tons**: `tons_port_q = Σ_m∈q tons_port_m`.
- **Deliverables:**

- `S1_port_month_tons.tsv` (unique `(port,year,month)`)
- `S1_port_quarter_tons.tsv` (unique `(port,year,quarter)`)
- `S1_terminal_month_tons.tsv` (unique `(port,terminal,year,month)`)
- `S1_qa.tsv`

QA-1 (must-pass): - Keys unique at their grains.
- No Eilat / All Ports rows.
- For months with terminal tons present, port-month tons equals the sum of terminals.

Stage 2 — Build TEU tables (from TEU mixed-frequency file only)

- **Monthly port TEU:** `Freq='Monthly'` & `Port ∈ {Ashdod, Haifa}`, keep only months ≤ `2021-08`.
- **Quarterly terminal TEU:** `Freq='Quarterly'` & `Port ∈ {Haifa, Haifa SIPG, Ashdod, Ashdod HCT}`, map to terminals.
- Build **port-quarter TEU**: `TEU_port_q = Σ_i TEU_i_q` per `(port,year,quarter)`.
- **Deliverables:**
- `S2_port_month_teu.tsv` (pre-reform monthly)
- `S2_terminal_quarter_teu.tsv` (post-reform quarterly)
- `S2_port_quarter_teu.tsv` (sum of terminals)
- `S2_qa.tsv`

QA-2 (must-pass): - Keys unique at their grains.
- For each `(port,year,quarter)` in terminal TEU, the port-quarter TEU equals the sum of terminals.
- No unexpected terminals after mapping.

Stage 3 — Load L_Proxy (terminal×month labor & Π)

- Standardize terminals via mapping; enforce `year, month` Int64; add `quarter` if missing.
- **Deliverables:**
- `S3_lproxy_clean.tsv` (selected columns only)
- `S3_qa.tsv`

QA-3 (must-pass): - Keys unique at `(port,terminal,year,month)`.
- Π available for expected terminal-years; basic coverage table printed by `(port,terminal,year)`.

Stage 4 — Build LP series

4A. Monthly port LP (Haifa & Ashdod, 2018-01..2021-08)

- Merge `S1_port_month_tons` with `S2_port_month_teu` for months ≤ `2021-08`.
- Compute `r_{p,m} = tons/TEU` → winsorize & rebase by `(port,year)` ⇒ `w_{p,m}`.
- Build monthly `Π_{p,m}` from `S3_lproxy_clean`: quarter-constant terminal shares × Π_{iy}.
- `LP_{p,m} = w × Π`; compute optional `LP_id = TEU / Σ_i L_hours`.

- **Deliverables:** `LP_Haifa_port_month.tsv`, `LP_Ashdod_port_month.tsv`, `S4A_qa.tsv`.

QA-4A (must-pass): - Keys unique.
- Annual preservation: within `(port,year)`, `|mean(LP) − mean(Π)| ≤ ε` for years with Π coverage.
- Winsorized `w` has mean 1 by `(port,year)`.

4B. Quarterly terminal LP (2021-Q3..2024-Q4)

- From `S1_port_quarter_tons` & `S2_port_quarter_teu`: `r_{p,q} = tons_port_q / TEU_port_q` → winsorize & rebase by `(port,year)` ⇒ `w_{p,q}`.
- For each terminal (Legacy/HCT/SIPG): `LP_{i,q} = w_{p,q} × Π_iy` (Π_{iy} from `S3_lproxy_clean` by terminal-year of `q`).
- **Deliverables:**
- `LP_Haifa_Legacy_quarter.tsv`
- `LP_Haifa_SIPG_quarter.tsv`
- `LP_Ashdod_Legacy_quarter.tsv`
- `LP_Ashdod_HCT_quarter.tsv`
- `S4B_qa.tsv`

QA-4B (must-pass): - Keys unique.
- `w` mean 1 by `(port,year)` on the quarterly groups.
- If `TEU_port_q ≤ 0` or missing → `w` NA → `LP` NA (and logged).
- Coverage table lists counts per terminal by year.

---

6) Error Handling & Guardrails

- **Drop** any row with `port='Eilat'` or `PortOrTerminal='All Ports'`.
- **No double-counting**: port-month tons via precedence rule (sum terminals if present).
- **TEU value** selection: use `TEU`; fallback to `1000 × TEU_thousands` only if `TEU` is missing.
- **DTypes**: keep `year, month` as `Int64`; `quarter ∈ {Q1..Q4}`; ratios as `float64`.
- **Key uniqueness** enforced at each stage; failure stops the pipeline and writes a small error sample to the stage QA.
- **Winsorization**: `[1%, 99%]` default (configurable), per `(port,year)`; **Rebase** makes mean(w)=1 within group.
- **NA logic**: ratios set to NA when numerator/denominator invalid; `LP` NA when either `w` or `Π` NA.

---

7) Deliverables & Naming

- Stage outputs: `S1_*.tsv`, `S2_*.tsv`, `S3_*.tsv`, `S4A_qa.tsv`, `S4B_qa.tsv`.
- Final six LP files named exactly as listed in Section 4.
- Optional: `LP_panel_long.tsv` stacking all six with `series_id` and `freq`.
- Per-run metadata in `_meta_lp_build.json` (row counts, parameter hash, input hashes).

---

8) CLI & Config (suggested)

```
python Build_LP_Panel.py
  --tons "Data/Output/monthly_output_by_1000_tons_ports_and_terminals.tsv"
  --teu  "Data/Output/teu_monthly_plus_quarterly_by_port.tsv"
  --lproxy "Data/L_proxy/L_Proxy.tsv"
  --out  "Data/LP"
  --cutover_month 2021-09
  --winsor_low 0.01 --winsor_high 0.99
```

---

9) Why this plan avoids prior pitfalls

- **No schema inference** from ambiguous columns (e.g., `Freq`, `MonthIndex`) — we read exactly what we need, with explicit filters.
- **No quarter inference** — we only use rows declared `Quarterly` for terminal TEU and build port quarters by summing terminals.
- **No double counting** of tons — port months are built from terminals when present.
- **Strict gates** at each stage — we never build LP on top of misaligned inputs.

This plan keeps the economics (LP = `w × Π`) intact and gives us **deterministic checkpoints** so we can debug issues immediately at their source before propagating them into the LP series.