Markov Chain Attribution + Budget Optimization — Deployment Design (Brief)

1) Purpose & Scope

Deploy a weekly batch pipeline that: - Builds **Order-1 Markov** attribution from customer journeys. - **Calibrates** a blended weight `opt_weight = α·removal_weight + (1−α)·provided_weight` (α via SMAPE minimization). - Solves a **linear program** with `Σ x == budget` under business/strategic constraints. - Publishes **recommendations & artifacts** for BI and stakeholders.

Primary KPI: predicted conversions at recommended mix
Secondary KPIs: uplift vs. current, attribution stability (MoM), runtime < 10 min.

---

2) Inputs & Outputs

Inputs (read-only, PostgreSQL `ads_db5.dw4`)

- `customer_journeys(customer_id, journey_step, channel, timestamp, converted, conversion_value, customer_segment, …)`
- `channel_performance(channel, current_monthly_spend, conversions_attributed, cost_per_conversion, min_monthly_spend, max_monthly_spend, conversion_rate_per_1000)`

Artifacts (written to storage; optional DB publish)

Path / Table	Description
`processed/transition_matrix.csv`	Absorbing Markov transition matrix (Order-1)
`processed/markov_attribution.parquet`	Removal effects + normalized weights (negatives floored to

	0)
`processed/channel_performance_enriched.parquet`	Performance + `opt_weight` (α-calibrated)
`processed/budget_scenario_{BUDGET}.parquet`	Optimal spend per channel for a given budget
`processed/sweep_alpha_budget.parquet`	Grid of predicted conversions across α × budget
(Optional) `analytics.budget_scenarios`	DB table for dashboards (scenario_id, budget, channel, optimal_spend, predicted_conversions, spend_change_pct)

3) Processing Logic

[Postgres ads_db5.dw4: customer_journeys (RO), channel_performance (RO)] → (SQLAlchemy/pandas) → [ETL & Attribution: normalize channels; build transition matrix (Order-1); baseline conversion via fundamental matrix; removal effects (floor negatives; normalize); α calibration (SMAPE vs historical)] → [Optimization (LP, CBC): objective = maximize Σ (x/1000)·conv_rate·opt_weight; constraints = Σ x == BUDGET; min ≤ x ≤ max per channel; strategic (Paid ≥ 30% budget; Social+Display ≤ 40%); outputs = optimal_spend, predicted_conversions, binding flags] → [Publish Artifacts: Parquet/CSV in ./processed/; optional upsert to analytics.budget_scenarios] → [Notify: Slack/Email summary (α*, uplift, bindings, runtime)]

4) Orchestration

- **Schedule:** Weekly (e.g., Monday 06:00 ET) after data closes.
- **Trigger:** Cron / Airflow / GitHub Actions.

Cron example (server time ET):

5) Implementation Notes

Runtime stack - Python 3.11; `pandas`, `numpy`, `sqlalchemy`, `psycopg2-binary`, `python-dotenv`, `pulp` (CBC). - Containerized (Docker) for reproducibility.

Secrets & Config - `.env` or Vault: `POSTGRES_USER`, `POSTGRES_PASSWD`, `POSTGRES_HOST`, `POSTGRES_PORT`, `POSTGRES_DATABASE`. - Config: `budgets`, `alpha_mode={"calibrated"|"fixed"}`, `alpha_fixed`, constraints toggles, notify webhook.

Processing steps 1. **Extract:** set `search_path=dw4`; read both tables to DataFrames (read-only role). 2. **Transform & Attribution:** - Title-case/normalize channels; build absorption matrix; compute baseline conversion. - Removal effect per channel; floor negatives to 0; normalize to weights. 3. **Calibration:** grid α∈[0,1]; minimize SMAPE between predicted @ current spend and `conversions_attributed`; set `alpha_best`. 4. **Optimization:** for each budget, solve LP with `Σ x == budget` and business bounds; tag binding constraints; compute predicted conversions. 5. **Publish:** write artifacts to `./processed/`; optional DB upsert to `analytics.budget_scenarios`. 6. **Notify:** summarize α*, uplift vs current, binding constraints, elapsed time.

---

6) Quality, Monitoring, Alerts

Data checks - Required columns present; `journey_step ≥ 1`. - Transition matrix row sums ~1.0 (±1e-6). - Baseline conversion ∈ [0,1]. - Negative removal effects captured & floored; list logged.

Calibration - Report α* and SMAPE; alert if SMAPE > 0.35.

Optimization - Solver status = Optimal; `Σ x == budget`. - Report **binding** min/max/strategic constraints.

Drift (optional) - MoM drift of weights per channel; alert if > 50% relative change.

Alerts - Job failure, SLA breach, calibration drift threshold, solver infeasible.

---

7) Security & Compliance

- Principle of least privilege: RO to `dw4`; separate RW user only for `analytics` (if used).
- Secrets via env/Vault; no secrets in logs; artifacts contain no PII.
- Access logs retained per org policy.

---

8) Rollback & Reproducibility

- Each run writes `processed/YYYYMMDD_HHMMSS/`.
- Store: git SHA, α*, budgets, constraints, solver log.
- Rollback by pointing BI/API to last good timestamp or DB snapshot.

---

9) Interfaces

BI/Dashboard - Read Parquet/CSV or `analytics.budget_scenarios`. - Standard visuals: Current vs Optimal spend, Predicted conversions, Binding constraints, Sensitivity (α × Budget).

(Optional) Read-only API - `GET /recommendation?budget=65000` → JSON: per-channel optimal_spend, predicted_conversions, α*, uplift, binding flags.