

Daniel Antares

Data Analyst · Data Scientist

Semarang, Indonesia (UTC+7) | daniel.antares@yahoo.com | [GitHub](#) | [LinkedIn](#)

Three end-to-end projects, each deployed and reproducible. Every figure is generated by committed code, and each project links to a live dashboard or documentation site you can open without installing anything. All repositories are public, with full READMEs covering methodology, data sources and stated limitations.

1 Data Job Market Intelligence

Which data skills are actually in demand, what do they pay, and how does that differ between Indonesia and the international remote market?

[▶ Read the write-up](#)

[Live dashboard \(wakes in ~30s\)](#)

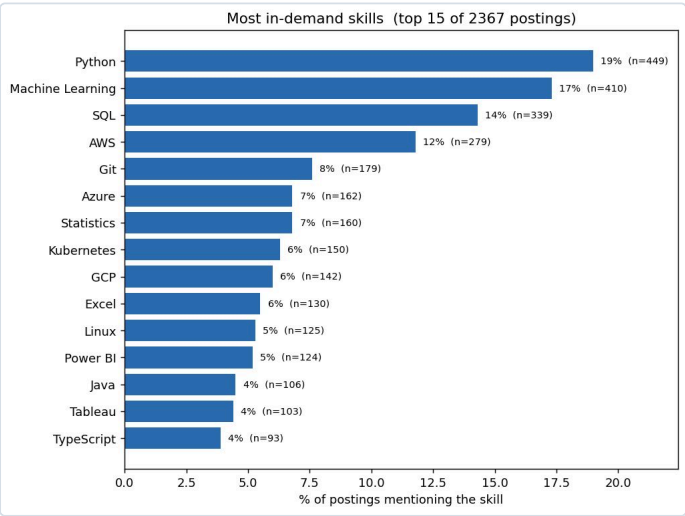
[GitHub](#)

WHAT I BUILT

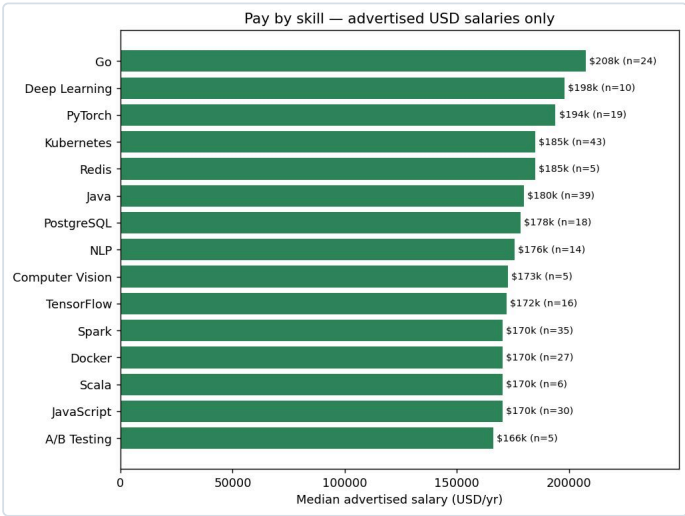
- An automated pipeline collecting data-job postings from **6 public APIs** (Remotive, Jobicy, RemoteOK, Adzuna, Jooble, JSearch) across **5 markets**, normalised onto a single schema and refreshed weekly.
- A **salary parser** turning free text ("¥80k - ¥100k", "\$18/hr") into structured values normalised to USD/year via a dated FX table, and a **skill extractor** over a curated ~70-skill taxonomy that resolves ambiguous names (R vs "R&D", Java vs JavaScript).
- A deduplicated Parquet warehouse, an interactive Streamlit dashboard with a live salary estimator, and a **weekly scheduled refresh** plus CI tests via GitHub Actions.

KEY FINDINGS (2,367 postings)

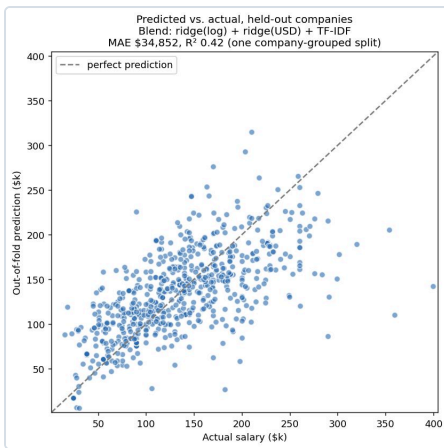
- **Local and remote markets want different things.** Indonesian roles lean analyst/BI: SQL **61%**, Python **58%**, Tableau **30%**. Malaysia leans ML-first (**43%**) with Power BI. Remote-international postings are Python and AWS heavy, with far more cloud and MLOps tooling.
- **Specialised and infrastructure skills carry a clear premium:** Deep Learning **\$198k**, PyTorch **\$194k**, Kubernetes **\$185k** against **\$120k to \$135k** for general analytics tooling. By role, ML Engineer **\$162k** leads Data Analyst **\$112k**.
- **The salary model went from R² 0.12 to 0.46 without changing the algorithm.** Every point came from data work: FX normalisation recovered 143 discarded rows, columns already in the warehouse (location, source, market) drove the single biggest jump, and mining the description text added the rest. Gradient boosting *lost* to a regularised linear model.



Most in-demand skills across all 2,367 postings.



Median advertised salary per skill, normalised to USD/year.



Predicted vs actual on held-out companies, so no employer appears in both train and test.

Tech: Python · pandas · PyArrow/Parquet · scikit-learn (ridge, TF-IDF, GroupKFold) · Streamlit · matplotlib · pytest · GitHub Actions (scheduled ETL + CI)

2 Brazilian E-Commerce Analytics: SQL to Power BI

What drives revenue, who are the best customers, and where do operations break down across a 100k-order marketplace?

[► Live Power BI dashboard](#)

[GitHub](#)

WHAT I BUILT

- Nine raw operational tables loaded into **PostgreSQL** and modelled into a documented **star schema**: two fact tables sharing one set of customer, product, seller and date dimensions, so metrics from either can be compared consistently.
- A library of **analytical SQL**: cohort retention, RFM segmentation, category Pareto and window-function trends, one business question per file, exposed to BI through seven reporting views.
- A four-page **Power BI** dashboard (Executive, Customers, Operations, Products & Payments) with ~15 DAX measures over the star schema.

KEY FINDINGS (~97.9k orders, R\$15.7M, 2017-01 to 2018-08)

- **The business runs on one-time buyers. 97% of customers purchased only once**, generating 94% of revenue, with under 1% repurchasing after the first month. The 3% who return are worth roughly **2x more** (R\$308 vs R\$161). That is the headline retention opportunity.
- **Revenue is concentrated.** RFM puts **16% of customers behind 31% of revenue**; São Paulo alone is 37% of revenue, and SP+RJ+MG together 62%.
- **Late delivery is the satisfaction killer.** 8% of orders arrive late and average **2.57★ against 4.29★** for on-time, a 1.7-star penalty. In promotional months the late rate itself climbs to **21%**.



Executive overview.



Customer segmentation and cohort retention.

Tech: PostgreSQL · advanced SQL (window functions, CTEs, NTILE) · dimensional modelling · Power BI + DAX · Docker · DuckDB

3 NYC Mobility: Analytics Engineering with dbt

Three ride-hail feeds with three different schemas, unified into one trusted fact table, with the data-quality tests that prove it.

► [Live warehouse documentation](#)

[GitHub](#)

HEADLINE FINDING

Two metering vendors report the same trip differently, and summing the published fare columns double-counts \$2.50 on 27% of all trips. One vendor folds congestion and airport surcharges into extra; the other reports them separately. On 2022 Q1 that is **2.34M trips overstated by about \$5.9M**. Nothing in the source schema warns you, and no chart would have looked wrong.

WHAT I BUILT

- A layered dbt warehouse over **9.28M trips**: 9 models (staging, ephemeral intermediate, marts), **70 data tests**, a SCD Type 2 snapshot, seeds, macros and an exposure, building in ~34 seconds on a laptop.
- Staging detects the vendor fare convention **per row rather than trusting VendorID**, because 120 rows contradict their own vendor's convention.
- Incremental monthly partition overwrite**, because the TLC restates published months, keying on the trip would append restatements as duplicates. Idempotency is asserted in CI by building twice and comparing row counts.
- Test severity chosen per test: **error** for our bugs, **warn** for upstream defects no transformation can fix, with bounded ranges picked from the observed distribution rather than round numbers.

WHAT THE TESTS CAUGHT

ISSUE FOUND BY A FAILING TEST	ROWS
Implausible trip distance	105,198
Voided or disputed trips	49,201
Implausible duration	10,038
Dropoff timestamped before pickup	8,739
Negative tips (3 with an undocumented payment code)	894

None are silently dropped. Each is flagged, counted, and surfaced as a column on the daily mart, so a broken upstream feed stays distinguishable from a real drop in demand.

SOURCES	STAGING	INTERMEDIATE	MARTS
yellow_tripdata_*	-> stg_tlc__yellow_trips		fct_trips (incremental)
green_tripdata_*	-> stg_tlc__green_trips	-> int_trips__	agg_trips_daily
fhvhv_tripdata_*	-> stg_tlc__fhvhv_trips	conformed	agg_zone_performance
taxi_zone_lookup	-> stg_tlc__zones		dim_zone, dim_date

Tech: dbt-core 1.9 · dbt-duckdb · DuckDB · dbt_utils · Python 3.12 · GitHub Actions (build, test, idempotency proof, docs deploy)