

Every record, accounted for.

How I built the tool an implementation engineer uses to move a new customer's messy data into a SaaS platform, and prove nothing was lost on the way.

Anwana Ntofon

The customer, the platform and every record are fictional and synthetic.

THE LEDGER

9,259	source rows in
7,163	landed in the platform
2,021	held back, each with a reason
75	set aside by the customer's rules
0	unaccounted for

A signed customer can't go live until their data does.

A B2B software company signs a new enterprise customer. Before the customer can use the platform, years of their data has to move in: accounts, contacts, subscriptions, history.

The data is never clean, and an import can report success while loading the wrong thing. It surfaces weeks later as billing errors and broken accounts.

Found before go-live: minutes to fix.

Found after go-live: Support, Engineering, Implementation and the customer, working out what changed and whether anything can be trusted.

WHERE IT GOES WRONG



Duplicates

the same company under two account numbers



Blanks and placeholders

required fields holding 'n/a' or nothing



Same name, other meaning

'status' of the account, or of the subscription?



Orphans

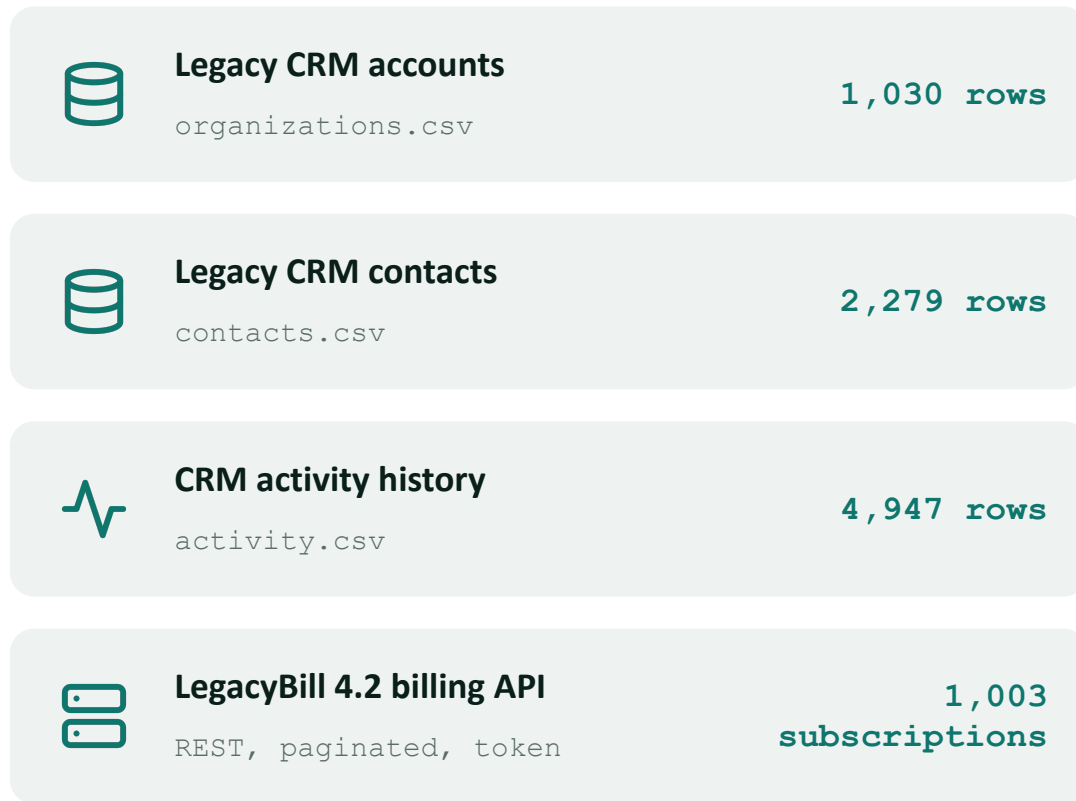
contacts pointing at accounts that do not exist



Silent success

the import worked; the counts don't add up

Apex Equipment Services: four sources, one deadline.



MERIDIAN

The SaaS platform the data must land in

- Strict schema and rules: one primary contact per account, no live subscription under an inactive account
- Reached only through its write API, like a real platform
- 14 business rules from the customer, in a document nobody had read in a year
- Go-live in three weeks

9,259 rows across four sources. All fictional, generated with the defects real exports have.

How this usually gets done.



01

Export to spreadsheets

Each system dumps what it has. Nobody measures it.



02

Map columns by eye

Guesses live in someone's head. No reasons, no review.



03

Import and hope

Success means the import ran, not that the data is right.



04

Hear from support

Billing errors and broken accounts, weeks later.

What's missing: a measurement of the data, a record of every decision and who made it, a rehearsal, and proof it landed exactly as sent.

What it had to do, and the lines it couldn't cross.

IT HAD TO

- ✓ Measure every source before anything moves
- ✓ Propose a home for every field, with a reason
- ✓ Put ambiguous calls in front of the customer
- ✓ Transform and validate with plain, tested code
- ✓ Rehearse the migration against the platform's API
- ✓ Prove every record is accounted for
- ✓ Answer the sponsor: ready, or what's blocking

WITHIN

-  **AI never touches a number**
no counting, validating or go/no-go
-  **No customer rows to the model**
statistics and shapes only
-  **Works with no AI at all**
a manual mode, nothing faked
-  **One week, one engineer**
a narrow workflow, finished
-  **Local first, then AWS**
EC2, S3 and an IAM role; no RDS

AI for meaning. Code for exactness.



THE MODEL MAY

- Propose where each column belongs, with a confidence and a reason
- Read the customer's rules written in plain English
- Write the questions only the customer can answer
- Explain why records are blocked, in plain language
- Draft the readiness summary from numbers it is given

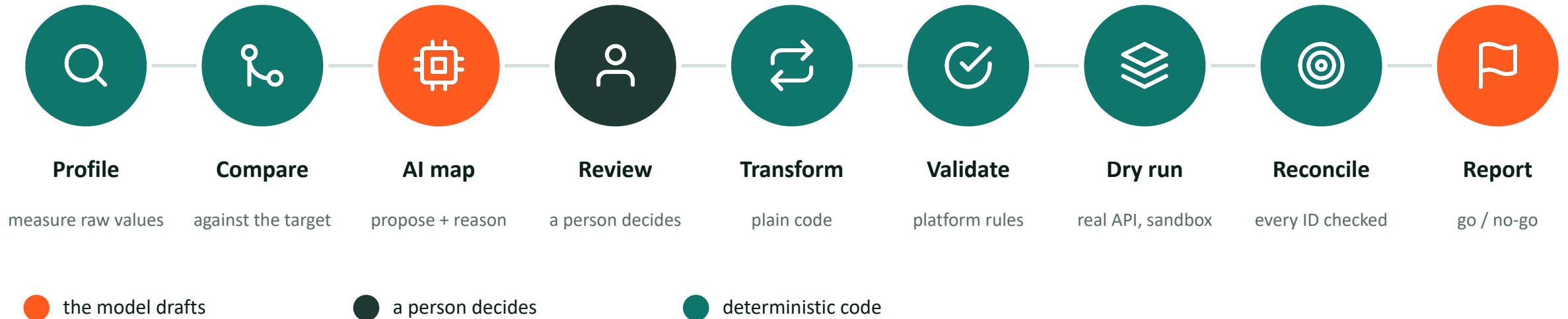


THE MODEL MAY NEVER

- Count rows or do arithmetic
- Validate a record or detect a duplicate
- Reconcile what was sent with what landed
- Decide that the customer is ready
- Approve its own suggestion

A model suggestion is a proposal a person reviews, never migration truth. No model-generated code is ever executed.

Nine stages, one record trail.



BUILT WITH

Python · FastAPI · PostgreSQL + Alembic · pandas · Pydantic · Streamlit · Typer · structlog · Docker Compose · Claude Opus 5 or GPT-5 behind one interface · GitHub Actions

Measure the data before moving any of it.



ROWS	COLUMNS	KEY COLUMN	DUPLICATE ROWS	ERRORS	WARNINGS
2,279	9	contact_ref 0 missing · 0 duplicated	0 exact copies	2 block migration	5 6 info
acct_num → organizations.csv.acct_num 2,182 resolve · 75 orphans (3.3%)					
Quality issues					
	COLUMN	WHAT	ROWS	SHARE	
error	contact_mail	106 contact_mail values are not valid emails (e.g. david.robertson@hammond-carrillo.com., ryan.king@stephanie.bell@wilson.com.)	106	4.7%	
error	acct_num	75 acct_num values point at a organizations.csv.acct_num that does not exist	75	3.3%	
warning	last_touch	last_touch arrives in 6 formats, 16.6% ISO 8601	1,808	83.4%	
warning	full_name	88 full_name values repeat once case and spacing are ignored	88	3.9%	
warning	phone_number	88 phone_number values are not valid phones (e.g. 979-9538, 500-3482, 282-7987)	88	4.0%	
warning	contact_mail	66 contact_mail values repeat once case and spacing are ignored	66	2.9%	
warning	last_touch	last_touch has dates in the future	8	0.4%	
info	phone_number	phone_number needs E.164 normalization	2,091	96.0%	

106

contact emails that are not valid addresses

75

contacts pointing at accounts that don't exist

6

date formats inside one column

Every finding is a count of cells that failed a stated rule, with examples, so the customer gets the exact list.

The AI proposes. A person decides.



SOURCE COLUMN	PROPOSED TARGET	CONFIDENCE	STATUS
acct_num	organization.organization_id	0.95	suggested
company_name	organization.name	0.98	suggested
company_status	organization.lifecycle_status	0.82	needs clarification
hq_state	organization.billing_region	0.65	needs clarification
customer_tier	organization.account_priority	0.45	needs clarification

0.45

customer_tier: it refuses to guess. The customer's own rule says tier is neither a plan nor a priority, so it writes a question for the customer instead.

0.85 below this line, nothing can be bulk-approved

100% of fields are decided by a person

Claude Opus 5's proposals for five of the account export's fifteen columns, as the reviewer sees them in the workbench.

Rehearse the migration against the real API.



Dry run · 2026-10-02 07:37:14 UTC

SOURCE ROWS	VALID	BLOCKED BY ERRORS	SKIPPED BY RULE	ACCEPTED BY TARGET	REFUSED OR FAILED
9,259	7,163	2,021	75	7,163	0
across every emitting source	308 with warnings	cannot migrate as they stand	the customer's own rules	of 7,163 sent	0 blocked

ENTITY	SOURCE ROWS	BUILT	SKIPPED	VALID	INVALID	WARNINGS	ATTEMPTED	ACCEPTED	REJECTED	FAILED	BLOCKED	RETRIES
organization	1,030	1,030	0	889	141	134	889	889	0	0	0	0
contact	2,279	2,279	0	1,702	577	152	1,702	1,702	0	0	0	0
subscription	1,003	1,003	75	414	514	22	414	414	0	0	0	0
activity	4,947	4,947	0	4,158	789	0	4,158	4,158	0	0	0	0

what the target platform holds in namespace `d09faa98-80c1-4d33-81cd-1fb3b9d685ab` : organization 889, contact 1,702, subscription 414, activity 4,158. 65.0s end to end.

reconciliation balanced 28 of 28 checks · every source row accounted for, the target holds exactly the accepted records · [open readiness](#)

7,163

sent, and every one accepted by the platform

28/28

reconciliation checks: every row accounted for

Each write carries the customer's own ID, so a retry after a timeout can never create a duplicate.

One answer for the sponsor: can we go live?



BLOCKED
Apex Equipment Services is BLOCKED for go-live.

Download markdown

Download json

SOURCE ROWS	IN SCOPE	LANDED	SHARE LANDED	BLOCKERS	CUSTOMER QUESTIONS
9,259	9,184	7,163	78.0%	4	7
followed through the rehearsal	75 set aside by rule	accepted by the target	of records in scope	1 condition	to send

BLOCKED

decided by stated rules over the measured facts, never by the model

7

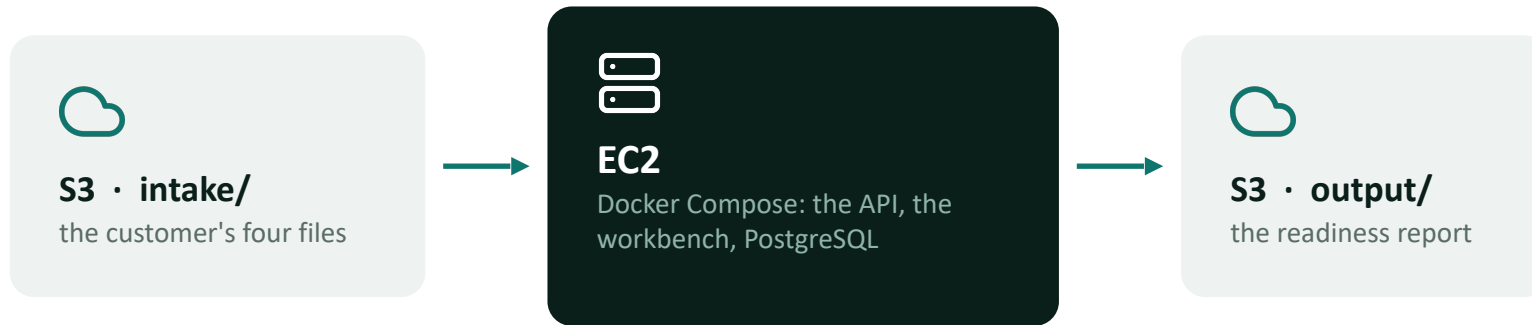
questions only the customer can answer, written out

WHAT STANDS IN THE WAY, IN THE REPORT'S WORDS

- 311** subscriptions are live against accounts that went dormant; the customer decides which side wins
- 215** activities use the kind 'Call', which the platform has no slot for; nothing is guessed
- 106** contacts have an email that isn't a valid address; the customer corrects them at the source

The model drafts the summary. Before anyone reads it, code rejects any number that isn't in the facts.

Same answer on AWS. No keys anywhere.



0

access keys on the server; the role hands it short-lived credentials

9,259

rows in, 7,163 accepted, 28/28 checks: identical to the local run

40 s

full rehearsal on EC2, against 65 s in local Docker



IAM role: read intake/, write output/. No delete, no other bucket, no access key.



Reached through an SSH tunnel: port 22 from one address. No app port on the internet.

```

upload failed: ./t.txt to s3://enterprise-onboarding-name-demo/intake/apex/t.txt An error occurred (AccessDenied) when calling the PutObject operation: User: arn:aws:sts::account-id:assumed-role/enterprise-onboarding-ec2/instance-id is not authorized to perform: s3:PutObject on resource: "arn:aws:s3::enterprise-onboarding-name-demo/intake/apex/t.txt" because no identity-based policy allows the s3:PutObject action
delete failed: s3://enterprise-onboarding-name-demo/output/apex/readiness-2026-10-03.md An error occurred (AccessDenied) when calling the DeleteObject operation: User: arn:aws:sts::account-id:assumed-role/enterprise-onboarding-ec2/instance-id is not authorized to perform: s3:DeleteObject on resource: "arn:aws:s3::enterprise-onboarding-name-demo/output/apex/readiness-2026-10-03.md" because no identity-based policy allows the s3:DeleteObject action
  
```

AWS refusing the role on purpose: a write into the customer's intake, and a delete. IDs covered.

What broke, and what I changed.



The rehearsal that lied

Against a dead platform, a rehearsal retried every record for 670 seconds, then called itself complete.

Fix A circuit breaker: 8.5 seconds, marked failed, with the reason.



Zero equals zero

When the platform couldn't be read back, reconciliation saw an empty target and balanced on nothing.

Fix An unreadable target now fails its own check, by name.



Keys inside the image

No .dockerignore: every Docker image carried the file with the AI provider keys.

Fix Fixed before anything was pushed. Image 4.27 GB down to 1.08 GB.



The cut-off answer

An AI answer truncated mid-JSON crashed the API with a 500.

Fix Now a clean error you can safely retry, caught by a test.

How I know it works.

449

automated tests: unit,
integration, end to end

95%

of the application code
exercised by them

16

failure scenarios driven for real,
each a row in the architecture
doc

CI

lint, types and the whole suite
on every push



Every screen driven in a real browser before it ships



Database migrations tested from empty, over real data, and
down and back up

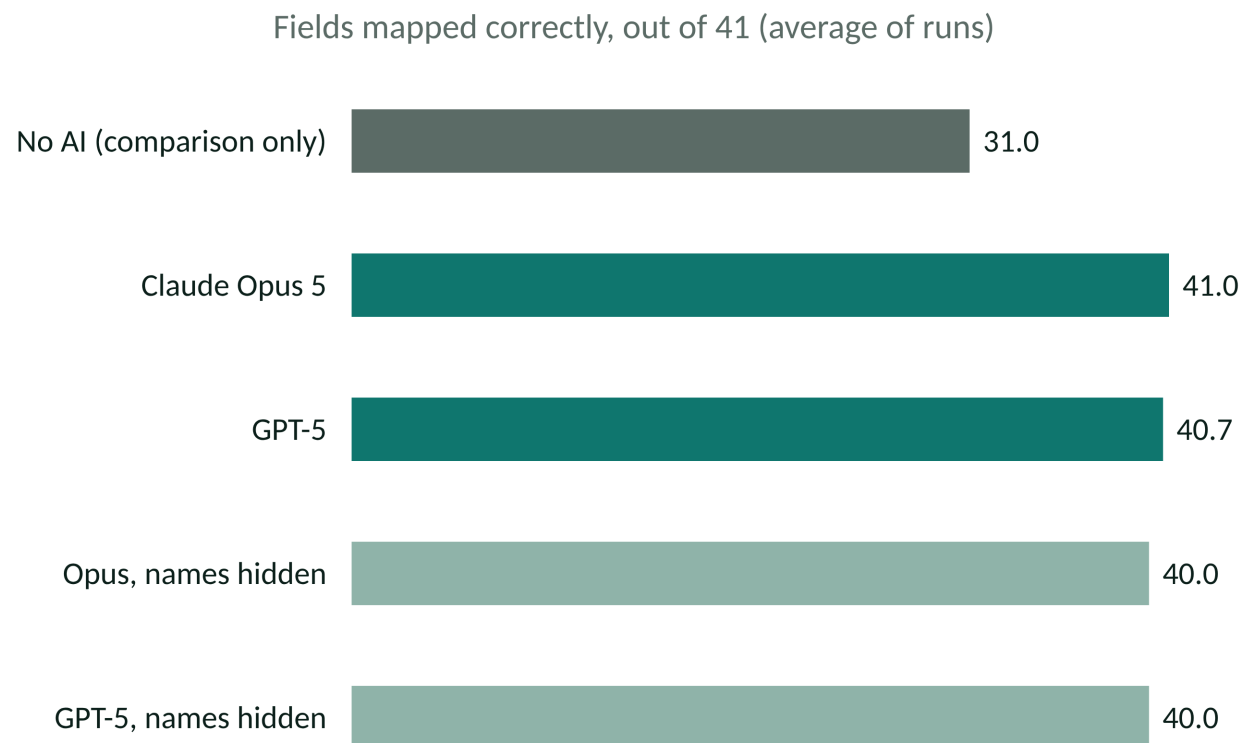


The logs are tested the way they are read: one ID finds every
step of a run



The AI evaluated against a golden set of 41 known answers, ten
runs

How the AI is wrong matters more than how often.



0

wrong answers at high confidence, across 10 runs, even with every column name hidden

The one miss came in at 0.62 and 0.68: below the approval line, flagged amber, in front of a person. The whole evaluation cost about \$6.

Measured on the synthetic customer.

MEASURED

9,259

source rows followed end to end

0

records refused by the platform

28/28

reconciliation checks balanced

8.5 s

to stop a rehearsal against a dead platform (was 670 s)

10.5 min

full rehearsal at ten times the volume, 72,302 records

\$6

the whole AI evaluation, both providers

ESTIMATED, NOT MEASURED

- Problems surface before anything moves, when they cost minutes, instead of after go-live.
- The customer gets an exact to-do list, not a surprise.
- Every decision has an owner and a reason when someone asks later.

Honest limits, and what production needs.

HONEST LIMITS

- One synthetic customer; a second customer with a different shape would make the numbers mean more
- The rehearsal is sequential and runs inside one request: fine at this size, ten minutes at ten times
- No login or roles yet: it's a single-tenant tool on a private network
- The summary check covers every number and the status, not every sentence of reasoning

BEFORE PRODUCTION

- Sign-in and roles: engineer, reviewer, the customer answering questions
- Runs on a worker queue, writes in parallel by account so answers don't change
- TLS everywhere and a secret manager instead of environment files
- Per-customer isolation, automated backups, tested restores
- On AWS: a load balancer in front instead of a tunnel, and infrastructure as code

THE WHOLE IDEA, IN ONE LINE

The AI proposes. A person decides. The code checks.

Built by Anwana Ntofon, a customer-informed engineer building AI tooling for the people who onboard and support customers.

AI

for meaning

Code

for exactness

People

for decisions

Proof

for every record