Voter Analytics Pipeline

Using Simulated Voter Records | Goodparty.org

Chris Adan

October 2025

Agenda

- Executive Summary
- Business Problem & Objective
- Technical Approach
- Orchestration & Lineage
- Data Quality & Governance
- Business Impact & Sample Analytics
- Surfacing Insights
- What's Next? Possible Extensions

Production-Ready Voter Analytics Platform

Scalable Data Ingestion and Transformation to Enable Deep Electoral Analytics

Task

- **Build a daily analytics pipeline**
- **Ingest raw voter records**
- **Curate analytics datasets for decision-making**

Project Scope: Take-home assessment

Timeline: 4-6 hours estimated

Tech Stack: Airflow | dbt | DuckDB | Streamlit

Automated

2 Python Processors
3 Airflow Dags
8 dbt Models
1 Streamlit Prototype

Quality

Statically Typed
Unit Tested
dbt Contracted
Auditable

Insights

5 Marts
4 Dims
2 Facts
Built for Analysis

From Voter Records to Campaign Strategy

Rich Electoral Analytics fueled by a pipeline built for growth

Challenges

Periodic voter file
(sub-weekly update)

No guaranteed validation

Integrity risk in input files

No historic election context

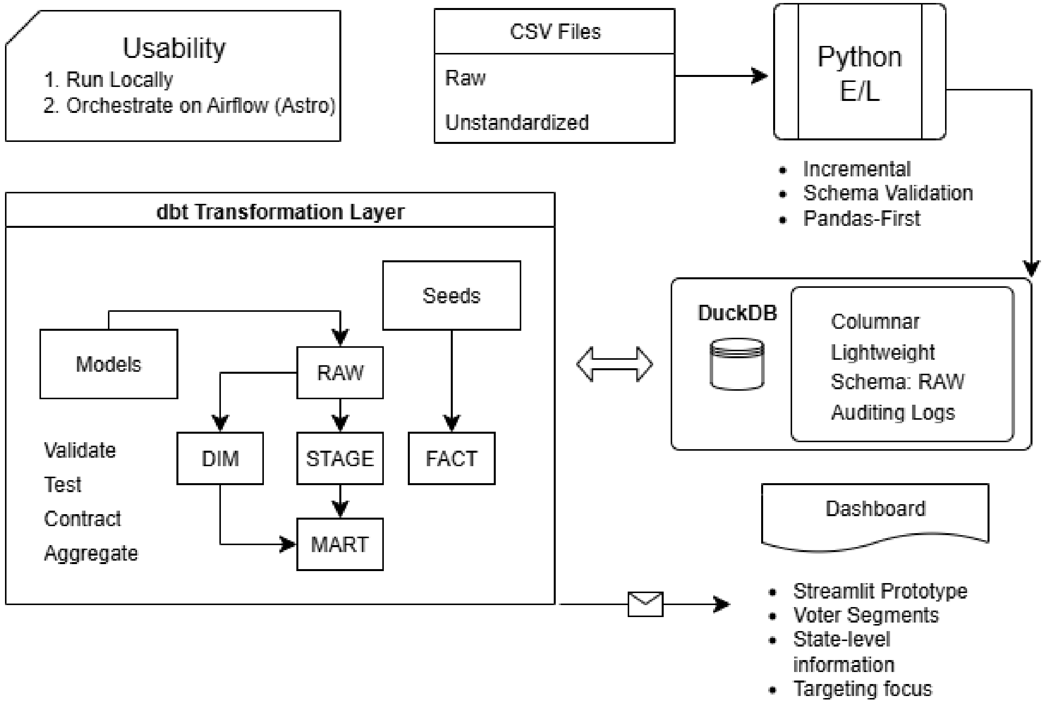
Required	Delivered	
Idempotency	✓	Automated EL-processor handling incremental ingestion
Intermediate	✓	Sanitized raw records in contracted DIM and STAGE layers
Mart	✓	Curated tables for core aggregations and targeting
Best Practice	✓	Modular, tested, documented, and reproducible

Extensions (Beyond MVP)	
Scalable Build System	3 DAGs | Setup > [Daily Pipeline, Monthly Seed]
Integrated Election Calendar	MIT Historic | Google Civic API | Federal Schedule
Behavioral Segmentation	6 engagement tiers, derived opportunity scoring
Production Patterns	Data contracting, custom macro(s)
Prototype Dashboard	Streamlit app surfacing 8 interactive visualizations

Technical Approach

Pandas ELT | Medallion Architecture

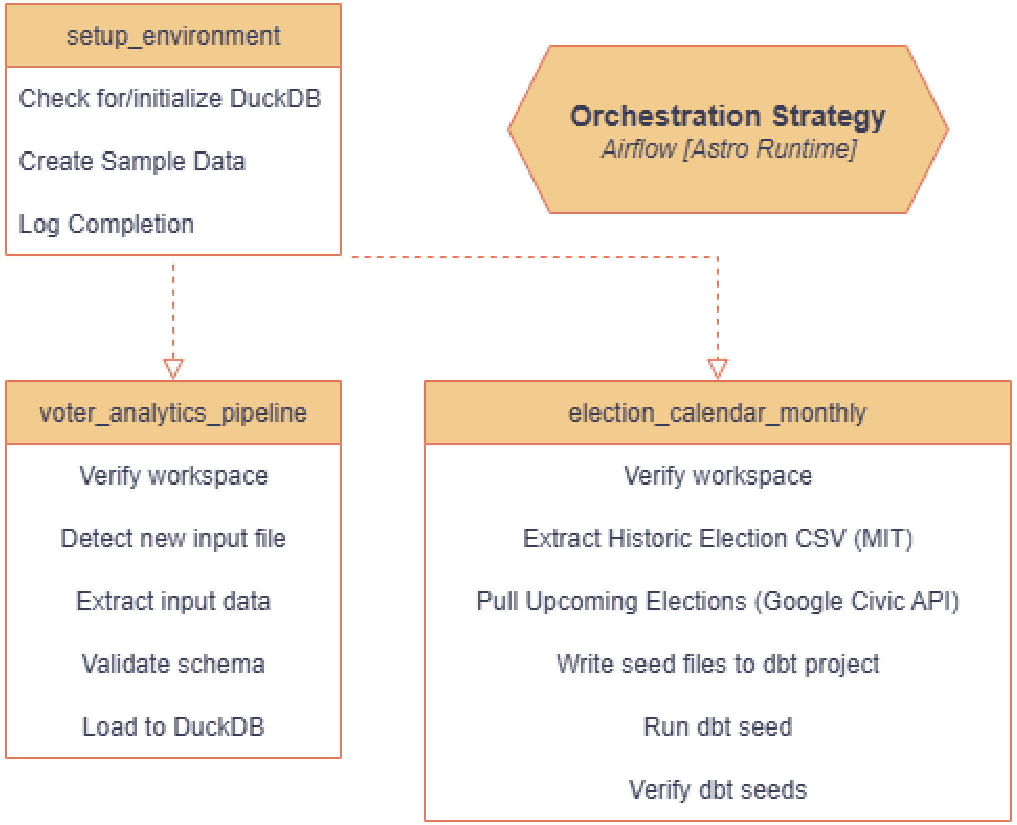
Layer	Technology	Rationale
Orchestration	Airflow (Astro) Cosmos	Rapid local development, dbt integration, portable
Ingestion	Python | Pandas	✅ **MVP: no-frills basic load strategy** 🚀 **Extension:** MD5 deduplication, schema validation, error thresholding, batch processing
Storage	DuckDB	Analytics-optimized, embedded, no infrastructure overhead, good for early phase
Transformation	dbt	✅ **MVP: essential dbt validation** 🚀 **Extension:** Ingestion unit testing, type-safe interfaces, dbt-expectations tests, macro-ready
Visualization	Streamlit + Plotly	Self-service analytics, no BI tool dependencies, good for prototyping



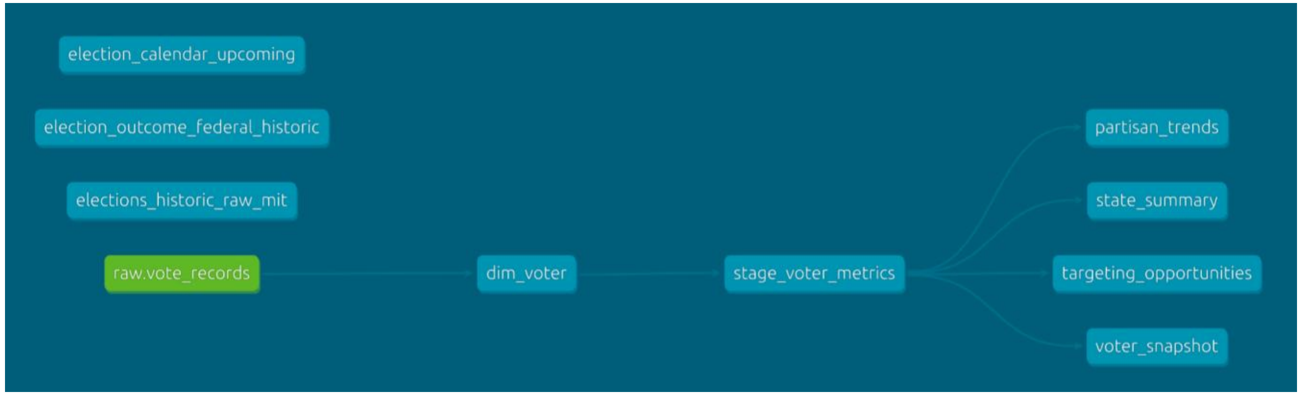

```
📊 Pipeline Stats:
- 3 Airflow DAGs
- 15+ dbt models (dim, stage, mart)
- 15+ data quality tests
- 4 production mart tables
- 2 election seed files (historic + upcoming)
```

Orchestration & Lineage

Layered approach for flexibility and reusability



dbt Lineage



Ready to leverage new seeds

1. DIM_VOTER: cleans and standardizes inbound vote_records upon landing in RAW
2. STAGE_VOTER_METRICS: prepares voter records for aggregation
3. Marts: partisanship, demographics, regional insight-ready

Data Quality & Governance

Production-Grade Testing, Contracts, Type Safety, and Referential Integrity at All Stages

Schema Enforcement
MVP: Validate key fields
Extension: type-safe EL pattern and dbt contracts

```yaml
# Example from voter_snapshot
contract:
  enforced: true
columns:
  - name: total_voters
    data_type: bigint
  - name: pct_current_voters
    data_type: decimal(5,2)
```

Quality Assurance
MVP: Tests for errors and warnings
Extension: EL unit-testing, dbt column-tests, dbt-expectations

Test Coverage	Covered Domains
DB Config + IO	Storage
Pre-ingest Typing	All inbound fields
Uniqueness	Voter IDs, State Codes
Null-checking	IDs, demographics, marts
Regex Validation	Emails, State Codes
Range Validation	Age (18-120), dates, percentages (0-100)
Accepted Values	Parties, States

ETL Safeguards
MVP: Handle Incremental Loads
Extensions:

Quick error check |
  >5% malformed records

Schema Validation |
  Enforces 10 expected fields

Record Deduplication |
  MD5 Hash:
  [ID, First/Last Name, Email]

Batch Process |
  Default 1,000 records

Garbage Collection |
  Closes connections
  Respects DuckDB 1-thread

Business Impact & Sample Analytics

Evolving Raw Records into Data Strategy

PROD_MART.VOTER_SNAPSHOT
Purpose: current voter composition
MVP: voter count by state, party
Extensions:
Behavioral segments | Engagement tiers

```
6 Engagement Segments:
- Current Voter (participated recently)
- Missed Last Election (lapsed once)
- Occasional Voter (2-3 lapses)
- Infrequent (4-6 lapses)
- Dormant (7+ lapses)
- Never Voted
```

Sample Insight:*
"Pennsylvania has 12,500 high-value 'Missed Last Election' target Democrats"

PROD_MART. TARGETING_OPPORTUNITIES
Purpose: ranked segments for GOTV campaigns
MVP: not required
Extensions:
Opportunity score algorithm prioritizing recency

```
Opportunity Score (0-100):
- 40% weight: Recent lapsers (1 election)
- 30% weight: Medium lapsers (2-3)
- 20% weight: Registration tenure
- 10% weight: Segment size
```

Sample Insight:
"Top 20 segments represent 45,000 recoverable voters with 78% recent engagement history"

PROD_MART.PARTISAN_TRENDS
Purpose: time series participation analysis
MVP: not required
Extensions:
Turnout trends over 9 election cycles (2008-2024)

Sample Insight:
"Independent voter participation dropped 18% from 2020 to 2022 midterms suggesting mobilization gap"

PROD_MART.STATE_SUMMARY
Purpose: geographic competitive landscape
MVP: voter count by state
Extensions:
Partisan lean classification | Engagement Opportunity Scoring

```
Partisan Lean Categories:
- Strong Dem/Rep (>10% margin)
- Lean Dem/Rep (5-10%)
- Competitive (2-5%)
- Highly Competitive (<2%)
```

Sample Insight:
"3 highly competitive states (NC, GA, AZ) have 35% recoverable voter populations"

<u>Strategic Impact</u>
1. Trend participation rates across cycles and partisanship
2. Identify high-priority re-engagement targets among core voter segments
3 .Prioritize competitive states for resource allocation
4. Segment voters for tailored messaging

** Example insights, not based on the provided sample data (n=~1.4k unique voters)*

Surfacing Insights

Interactive Streamlit dashboard prototype, no SQL required

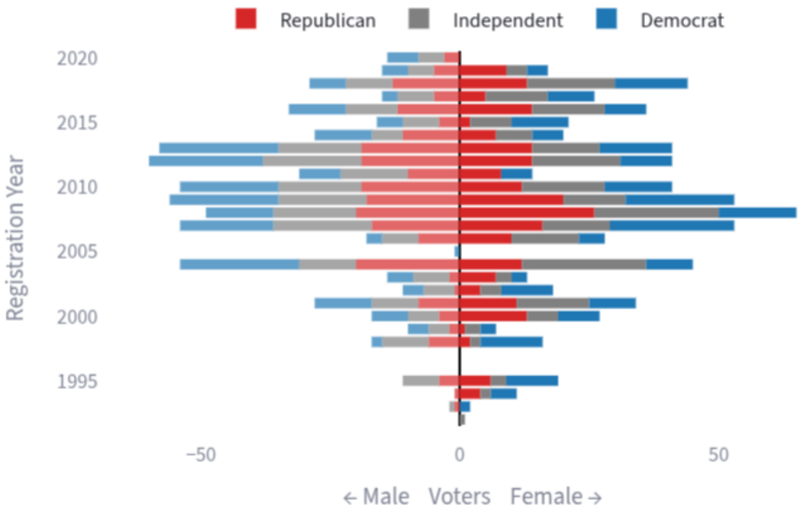
🎯 Top Targeting Opportunities

State	Age Group	Party	Opportunity Score
WA	50-64	Republican	56.7%
OR	30-49	Independent	56.7%
NH	30-49	Democrat	56.7%
AK	30-49	Independent	53.4%
PA	30-49	Republican	53.4%
OR	50-64	Democrat	53.4%
NH	50-64	Republican	53.4%
NJ	30-49	Democrat	47.5%
DE	30-49	Democrat	47.5%
CA	30-49	Democrat	47.5%

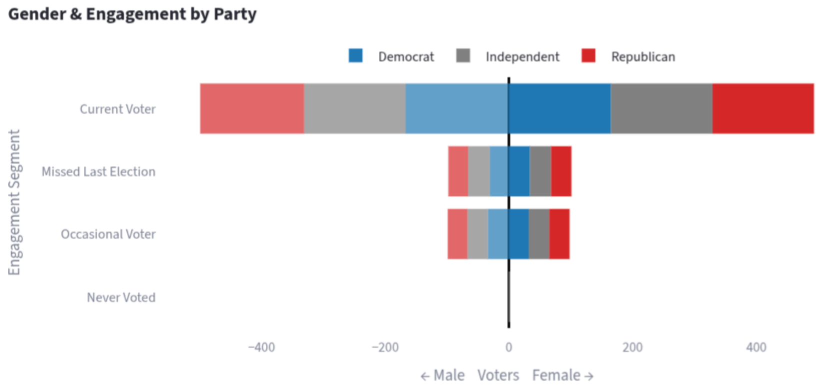
🎨 Dashboard Features:
- 8 interactive charts (bar, line, heatmap, diverging, pie)
- Real-time filtering (state, party, engagement tier)
- Drill-down from state → demographic segments
- Export-ready tables for campaign teams

Voter Registrations by Year

Partisanship by Gender Over Time



🎯 Targeting by Engagement & Demographics



What’s Next?

Scaling the platform from MVP to enterprise

Phase 1 [Complete]

- Idempotent ETL pipeline
- Medallion architecture

  (raw->dim->stage->mart)
- Quality testing and data contracts
- Behavioral segmentation and opportunity scoring
- Interactive dashboard
- Portable distribution

Potential Extensions

- **Phase 2**: Production Hardening **[2-4 weeks]**
  - Refactor/integrate into existing platform (DB, BI, docs)
  - Quality Improvements: Deeper testing, automated Airflow monitoring (Slack/email), audit suite
  - Performance improvements: partitioning, snapshotting (e.g. address changes), query optimizations (clustering, indexes, materialized views)
  - Security & Compliance: PII hashing, RBAC, privacy compliance
- **Phase 3**: Advanced Analytics **[2-3 months]**
  - Predictive modeling: turnout prediction, churn risk modeling
  - Enriched dimensions: household clustering, social listening
  - Real-Time Operations: automate anomaly detection. CDC from upstream systems

Thank You | Discussion

How does the team balance velocity and technical rigor when making architectural decisions?
Framework for evaluating build-vs-buy and ‘good enough’ vs. production-hardened tradeoffs?

What are the biggest data quality challenges the team is facing today?
How do you handle schema evolution in production?
How does the team approach net-new data models? (For example, Serve)
When something breaks in production, what is the incident response process?

What strategic or operational decisions are hardest to make with data you have today?
How do you balance data investments between Win (mature) and Serve (emerging)?

Materials

GitHub Repository

PEW Research Party Affiliation Fact Sheet (NOPRS)

Socials

Find me on LinkedIn

Check out my GitHub

Read on Medium