Architecture Roadmap: The Unified Engine Migration

Version: 1.2.0 **Date:** 2025-11-24 **Status:** Active - Phase 1 (Data Decoupling)

---

1. The Strategic Vision

Goal: Create a single analysis engine that powers both a Command-Line Interface (CLI) and a future Web Interface (Django). **Core Constraint:** "Write Once, Render Everywhere." Logic must not be duplicated between CLI and Web. **Output Standard:** "The Portable Artifact." Reports are self-contained HTML5 files with embedded Data (JSON), Logic (JS), and Styling (CSS) that function fully offline.

2. Architectural Decision Records (ADRs)

ADR-001: Data Abstraction Layer (DAL)

- **Decision:** All business logic (sums, rates, set intersections) must be extracted from Report classes into `data_aggregators/`.
- **Constraint:** Aggregators must return **Pure Python Primitives** (Dict/List). No Pandas objects allowed in return values.
- **Reason:** Ensures the data is JSON-serializable for the Web API and embedded HTML.

ADR-002: Unified Visualization (Plotly + Jinja2)

- **Decision:** Replace Matplotlib with **Plotly (Python)**. Use **Jinja2** for HTML generation.
- **Reason:** Plotly generates interactive JS automatically. Jinja2 separates structure from code.
- **Export:** "Publication Quality" static files (.png) will be generated via Server-Side Export (Kaleido), not client-side screenshots.

ADR-003: Slice-Based Aggregation

- **Decision:** Aggregators are stateless. They accept filtered DataFrames (slices) rather than filtering internally based on rigid "Band" loops.
- **Reason:** Supports flexible scoping (e.g., mixed-mode contests, global scope contests like Sweepstakes).

ADR-004: Quality Rigor (Workflow v2.4.0)

- **Decision:** All development must adhere to `AIAgentWorkflow.md` v2.6.0+.

- **Constraints:**
  - **Pre-Flight Checklists** are mandatory before code generation.
  - **Systemic Analysis** is mandatory for bug fixes.
  - **Verification Commands** must be provided at the end of every Builder session.

---

3. Master Transition Timeline

Phase 1: Data Decoupling (Current Phase)

- **Goal:** Move logic from `reports/` to `data_aggregators/`.
- **Status:**
  - ☒ Batch 1: Base Infrastructure & Stats.
  - ☒ Batch 2: Time Series (Rate Sheets).
  - ☒ Batch 3: Matrix Data (Heatmaps).
  - ☒ Batch 4: Categorical & Comparison.
  - ☐ **Batch 5: Validation (Refactor WAE Oddities) (Active).**

Phase 2: Visualization Standardization

- **Goal:** Replace Matplotlib with Plotly/Jinja2.
- **Tasks:**
  - Create `report_template.html`.
  - Refactor reports to output Plotly JSON.
  - Implement "Offline Mode" (embed JS) vs "Web Mode" (CDN).

Phase 3: The Web Foundation

- **Goal:** Initialize Django.
- **Tasks:**
  - Setup Django project.
  - Create Views that import `ContestLog` and `Aggregators`.
  - Serve the HTML artifacts generated in Phase 2.

Phase 4: Advanced Features

- **Goal:** Configuration & Export.
- **Tasks:**
  - Implement Server-Side Export (Kaleido).
  - Upgrade Regression Test Harness to compare JSON data blobs.

---

4. Current State (Handoff Note)

- **Date:** 2025-11-24

- **Context:** We are executing **Phase 1, Batch 5**.
- **Immediate Goal:** Implement `WaeStatsAggregator` to handle WAE-specific logic (Weighted Multipliers, QTCs) and refactor the two WAE reports.
- **Constraint:** Do NOT start Phase 2 (Plotly) yet. Finish the DAL using existing text renderers to verify data integrity first.