

Implementation Plan: Data Abstraction Layer (Phase 2 - Batch 4)

Version: 1.6.1 **Date:** 2025-11-24 **Status:** Execution (Phase 1 - Batch 4: Categorical)

1. Objective

Implement `CategoricalAggregator` to centralize logic for Point Breakdowns and Set Comparisons. Refactor dependent reports. **Constraint:** The Public API of the Aggregator must NOT accept or return Pandas objects. It must accept Domain Objects (`ContestLog`) or primitives, and return primitives (Dicts).

2. Component Design

2.1. New Module: `contest_tools/data_aggregators/categorical_stats.py`

Class: `CategoricalAggregator`

- **Method:** `get_points_breakdown(logs: List[ContestLog], band_filter: str=None, mode_filter: str=None) -> Dict`
 - **Input:** List of Log objects and optional filter strings.
 - **Logic:** Internally extract `get_valid_dataframe()`, apply filters, and aggregate points.
 - **Return:** Dict of scalars and breakdowns (Pure Python).
- **Method:** `compute_comparison_breakdown(log1: ContestLog, log2: ContestLog, band_filter: str=None, mode_filter: str=None) -> Dict`
 - **Input:** Two Log objects and optional filter strings.
 - **Logic:**
 1. Internally extract DataFrames from logs.
 2. Apply Band/Mode filters.
 3. Determine Uniqueness Keys based on filters (e.g., if Mode filter is active, key is ['Call']; if 'All Modes', key includes ['Call', 'Mode']) to handle mixed-mode collisions).
 4. Perform Set Intersection.
 - **Return Schema (Pure Python):**

```
{  
    "log1_unique": { "run": int, "sp": int, "unk": int },  
    "log2_unique": { "run": int, "sp": int, "unk": int },  
    "common": { "run_both": int, "sp_both": int, "mixed": int },  
    "metrics": { "total_1": int, "total_2": int, "unique_1": int, "unique_2": int,  
}
```

2.2. Refactoring Targets

- `chart_point_contribution.py & chart_point_contribution_single.py`
 - Update `_create_plot_for_band`.
 - Call `agg.get_points_breakdown(self.logs, band_filter=band)`.
- `chart_qso_breakdown.py` (Stacked Bar Chart)
 - Instantiate `CategoricalAggregator`.
 - Loop bands.
 - Call `agg.compute_comparison_breakdown(log1, log2, band_filter=band)`.
 - Map the returned dictionary keys to the chart columns.
- `html_qso_comparison.py` (HTML Table)
 - Refactor `_aggregate_data`.
 - Loop bands.
 - Call `agg.compute_comparison_breakdown(log1, log2, band_filter=band)` for each band row.
 - Call `agg.compute_comparison_breakdown(log1, log2)` (no filters) for the "All Bands" row.

3. Verification

- **Script:** `run_regression_test.py`
- **Criteria:** Zero regressions.

4. Builder Manifest

```
manifest.txt
contest_tools/data_aggregators/categorical_stats.py
contest_tools/reports/chart_point_contribution.py
contest_tools/reports/chart_point_contribution_single.py
contest_tools/reports/chart_qso_breakdown.py
contest_tools/reports/html_qso_comparison.py
run_regression_test.py
```