Technical Specification: Wrought Iron (WI)

Version 1.0.0 – "The Air-Gapped Foundry"

Classification: Enterprise Offline DataOps Platform **Deployment Target:** High-Security / Air-Gapped Local Environments **Core Stack:** Python 3.11+, Typer, Pandas, Textual, SQLite

---

1. Product Architecture

1.1 Core Philosophy

- **Zero Egress:** The application must never initiate an outbound network request. All dependencies (mapping tiles, tokenizer models) must be bundled locally.
- **Universal Ingest:** The tool must accept any SQLite file (`.db`) and automatically detect and flatten JSON columns (the "Backpack" pattern) into queryable virtual columns.
- **Deterministic Forensics:** Every data mutation must be reversible via snapshots and logged in a cryptographically verifiable audit trail.

1.2 The Technology Stack

- **CLI Framework:** `Typer` (Standardized Subcommand Architecture)
- **TUI Framework:** `Textual` (for Module 9 "Command Center")
- **Data Engine:** `Pandas` (Default) & `Polars` (High-Performance Flag via `--engine polars`)
- **Storage:** `SQLite3` (Target Data) & `DuckDB` (Analytical Cache)
- **Intelligence:** `Scikit-learn` (Imputation), `Rapidfuzz` (String Matching), `Microsoft Presidio` (PII)
- **Visualization:** `Plotext` (Terminal), `Sweetviz` (HTML Exports)
- **Security:** `Cryptography` (Fernet AES-256), `ReportLab` (PDF Generation)

---

2. Functional Specification (The 12-Module Matrix)

Module 1: Infrastructure (`wi connect`)

Responsibility: Connection Lifecycle & Storage Management

1. `wi connect file [PATH]` - Register an existing `.db` file as the active target.
2. `wi connect new [PATH]` - Initialize a fresh SQLite database with the standard WI metadata tables.
3. `wi connect list` - Display a history of recently accessed databases with file size and last-access timestamps.
4. `wi connect alias [NAME] [PATH]` - Assign a persistent short-name (e.g., `prod`) to a file path.

5. `wi connect info` - Display low-level metadata: Page size, WAL mode status, encoding, and user permissions.
6. `wi connect vacuum` - Execute `VACUUM` to rebuild the DB file and reclaim disk space from deleted rows.
7. `wi connect merge [TARGET] [SOURCE]` - ETL command to merge two separate SQLite files into one.
8. `wi connect encrypt [PATH]` - Encrypt the entire database file at rest using AES-256 (User prompted for password).
9. `wi connect decrypt [PATH]` - Decrypt a WI-encrypted file for use.
10. `wi connect integrity-check` - Run `PRAGMA integrity_check` to detect corruption in the binary file.

Module 2: Schema Management (`wi schema`)

Responsibility: Structural Introspection & Evolution

1. `wi schema list` - List all tables, views, and indexes.
2. `wi schema describe [TABLE]` - Show column names, inferred types, and primary/foreign keys.
3. `wi schema inspect [TABLE]` - **Profile:** Calculate non-null %, unique %, and top 5 most common values per column.
4. `wi schema diff [TABLE_A] [TABLE_B]` - Comparative analysis of two schemas (useful for checking drift).
5. `wi schema graph` - Generate a text-based Entity Relationship Diagram (ERD) of foreign keys.
6. `wi schema detect-json [TABLE]` - Scan `TEXT` columns to identify valid JSON structures (The "Backpack" scan).
7. `wi schema flatten [TABLE] [COL]` - Permanently explode a JSON column into distinct first-class columns.
8. `wi schema rename-col [TABLE] [OLD] [NEW]` - Safely rename a column (handling SQLite limitations).
9. `wi schema drop-col [TABLE] [COL]` - Remove a column and reconstruct the table.
10. `wi schema cast [TABLE] [COL] [TYPE]` - Force type conversion (e.g., String -> Integer) with error handling policies.

Module 3: Data Exploration (`wi query`)

Responsibility: Retrieval & Filtering

1. `wi query head [TABLE] -n [INT]` - Display the first N rows.
2. `wi query tail [TABLE] -n [INT]` - Display the last N rows.
3. `wi query filter [TABLE] --where "[PANDAS_QUERY]"` - Filter rows using pythonic syntax (e.g., `age > 18 and city == 'Melbourne'`).
4. `wi query sql "[QUERY]"` - Execute raw SQL directly against the connection.
5. `wi query search [TABLE] [TERM]` - Perform a global full-text search across all columns.
6. `wi query sort [TABLE] [COL] --asc/--desc` - Output sorted dataset.
7. `wi query sample [TABLE] --frac [FLOAT]` - Return a random statistical sample.
8. `wi query distinct [TABLE] [COL]` - List unique values for a specific field.

9. `wi query find-nulls [TABLE]` - Return rows containing `NULL` in any column.
10. `wi query dups [TABLE] [COL]` - Identify rows with duplicate values in the specified column.

Module 4: Analytics (`wi aggregate`)

Responsibility: Statistical Computing

1. `wi aggregate groupby [TABLE] [GROUP_COLS] --agg [DICT]` - Pandas-style GroupBy (e.g., `{'age': 'mean'}`).
2. `wi aggregate pivot [TABLE] [INDEX] [COL] [VAL]` - Create a Pivot Table.
3. `wi aggregate describe [TABLE]` - Generate summary statistics (Mean, Std, Min, Max, Quartiles).
4. `wi aggregate corr [TABLE]` - Generate Pearson/Spearman correlation matrix.
5. `wi aggregate skew [TABLE]` - Calculate distribution skewness for numeric columns.
6. `wi aggregate kurtosis [TABLE]` - Calculate distribution kurtosis (outlier propensity).
7. `wi aggregate moving-avg [TABLE] [COL] --window [INT]` - Calculate rolling averages.
8. `wi aggregate rank [TABLE] [COL]` - Generate dense ranks for a column.
9. `wi aggregate bin [TABLE] [COL] --bins [INT]` - Discretize continuous variables into buckets.
10. `wi aggregate crosstab [TABLE] [COL_A] [COL_B]` - Generate frequency matrix between two categorical variables.

Module 5: Visualization (`wi plot`)

Responsibility: Terminal-Based Charting (Plotext)

1. `wi plot bar [TABLE] [CAT_COL] [NUM_COL]` - Vertical Bar Chart.
2. `wi plot barh [TABLE] [CAT_COL] [NUM_COL]` - Horizontal Bar Chart (Best for long labels).
3. `wi plot hist [TABLE] [NUM_COL] --bins [INT]` - Frequency Histogram.
4. `wi plot scatter [TABLE] [X] [Y]` - Scatter Plot for correlation checking.
5. `wi plot line [TABLE] [X] [Y]` - Line Chart for time-series.
6. `wi plot box [TABLE] [NUM_COL]` - Box-and-Whisker plot (Outlier visualization).
7. `wi plot matrix [TABLE]` - Pairplot/Scatter Matrix of all numeric columns.
8. `wi plot heatmap [TABLE] [COL_A] [COL_B]` - 2D Density Heatmap.
9. `wi plot save [PATH]` - Export the last generated chart to PNG/SVG.
10. `wi plot theme [NAME]` - Toggle color scheme (Dark/Light/High-Contrast).

Module 6: Data Wrangling (`wi clean`)

Responsibility: Repair & Imputation

1. `wi clean impute-mode [TABLE] [COL]` - Fill `NaN` with the most frequent value.
2. `wi clean impute-group [TABLE] [TARGET] [GROUP_COL] --std-max [FLOAT]` - **Cohort Imputation:** Fill missing values based on group mode, *aborting* if group variance exceeds `std-max` (Safety Valve).
3. `wi clean ml-impute [TABLE] [COL] --neighbors [INT]` - **KNN Imputation:** Use K-Nearest Neighbors to predict missing values.
4. `wi clean dedupe [TABLE] [COL] --threshold [INT]` - Interactive fuzzy deduplication session.

5. `wi clean harmonize [TABLE] [COL]` - Cluster similar text variations (e.g., "USA", "U.S.A.") and standardizing.
6. `wi clean regex-replace [TABLE] [COL] [PATTERN] [REPL]` - Advanced string substitution.
7. `wi clean drop-outliers [TABLE] [COL] --sigma [FLOAT]` - Nullify values exceeding N standard deviations.
8. `wi clean map [TABLE] [COL] [FILE]` - Apply a CSV-based dictionary mapping/lookup.
9. `wi clean trim [TABLE] [COL]` - Remove leading/trailing whitespace.
10. `wi clean validate-schema [TABLE] [RULES_FILE]` - Check data against a JSON schema definition.

Module 7: Geospatial (`wi geo`)

Responsibility: Spatial Analysis

1. `wi geo validate [TABLE] [LAT] [LON]` - Validate coordinates fall within global bounds.
2. `wi geo geocode [TABLE] [ADDR_COL]` - **Offline Geocoding:** Map Suburb/Postcode to Lat/Lon using a local bundled lookup table.
3. `wi geo reverse [TABLE] [LAT] [LON]` - Convert coordinates to Suburb name.
4. `wi geo distance [TABLE] [LAT] [LON] --target [LAT] [LON]` - Calculate Haversine distance.
5. `wi geo cluster [TABLE] --eps [FLOAT]` - **DBSCAN:** Group records into spatial clusters based on density.
6. `wi geo centroid [TABLE]` - Calculate the geometric center of the dataset.
7. `wi geo bounds [TABLE]` - Return the North/South/East/West bounding box.
8. `wi geo heatmap [TABLE]` - Render an ASCII density map in the terminal.
9. `wi geo nearest [TABLE] --target-file [POI_DB]` - Find nearest Point of Interest (e.g., Hospital) for each row.
10. `wi geo export-geojson [TABLE]` - Export data to standard `.geojson` for GIS software.

Module 8: Machine Learning (`wi ml`)

Responsibility: Predictive Modeling

1. `wi ml train-classifier [TABLE] [TARGET] [FEATURES]` - Train a Random Forest Classifier.
2. `wi ml train-regressor [TABLE] [TARGET] [FEATURES]` - Train a Linear/Ridge Regressor.
3. `wi ml predict [TABLE] [MODEL_PATH]` - Apply a saved model to fill empty columns.
4. `wi ml score [TABLE] [MODEL_PATH]` - Calculate Accuracy/R2/F1 Score.
5. `wi ml feature-importance [MODEL_PATH]` - List which columns drive the prediction.
6. `wi ml save-model [PATH]` - Serialize model to `.pkl`.
7. `wi ml load-model [PATH]` - Hydrate model from disk.
8. `wi ml cluster-kmeans [TABLE] --k [INT]` - Unsupervised K-Means clustering.
9. `wi ml detect-anomalies [TABLE]` - **Isolation Forest:** Flag statistical outliers.
10. `wi ml split [TABLE] --train [FLOAT]` - Split dataset into Train/Test subsets.

Module 9: Audit & Security (`wi audit`)

Responsibility: Compliance & Forensics

1. `wi audit log-view` - Display the internal Wrought Iron immutable audit log.
2. `wi audit hash-create [TABLE]` - Generate SHA-256 fingerprint of the table state.
3. `wi audit hash-verify [TABLE] [HASH]` - Verify current state matches the fingerprint.
4. `wi audit snapshot [TABLE]` - Create a named restore point.
5. `wi audit rollback [TABLE] [SNAPSHOT_ID]` - Revert data to a previous state.
6. `wi audit scan-pii [TABLE]` - **Presidio:** Scan for Credit Cards, Phones, Emails in text fields.
7. `wi audit encrypt-col [TABLE] [COL] --key [FILE]` - Column-level encryption.
8. `wi audit decrypt-col [TABLE] [COL] --key [FILE]` - Column-level decryption.
9. `wi audit anonymize [TABLE] [COL]` - Apply masking (e.g., `J*** D**`).
10. `wi audit export-cert` - Generate a PDF "Chain of Custody" certificate.

Module 10: Operations (`wi ops`)

Responsibility: Automation

1. `wi ops schedule create [CMD] --cron [STR]` - Register a command to the internal scheduler.
2. `wi ops schedule list` - View active automated tasks.
3. `wi ops schedule delete [ID]` - Remove a task.
4. `wi ops pipeline run [YAML_FILE]` - Execute a multi-step workflow defined in YAML.
5. `wi ops trigger [ID]` - Force run a scheduled task immediately.
6. `wi ops logs` - View execution history of background jobs.
7. `wi ops drift-check [TABLE] --baseline [SNAPSHOT]` - Automated Quality Control check.
8. `wi ops alert-config --email [ADDR]` - Configure local hooks for failure alerts.
9. `wi ops maintenance` - Trigger DB optimization/Index rebuilding.
10. `wi ops export-status` - Output system health JSON for external monitoring.

Module 11: Collaboration (`wi collab`)

Responsibility: Knowledge Sharing

1. `wi collab view save [NAME] --query [SQL]` - Save a complex filter as a Virtual View.
2. `wi collab view list` - List available views.
3. `wi collab view load [NAME]` - Load a view into the current workspace.
4. `wi collab config export [CMD] [FILE]` - Export parameters (e.g., Regex patterns) to JSON.
5. `wi collab config import [FILE]` - Import shared parameters.
6. `wi collab recipe bundle [NAME]` - Zip up Views + Configs + Maps into a `.wi` bundle.
7. `wi collab recipe install [FILE]` - Install a team bundle.
8. `wi collab notes add [TABLE] "[TEXT]"` - Add metadata annotation to a table.
9. `wi collab notes show [TABLE]` - Read annotations.
10. `wi collab workspace dump` - Export full environment state.

Module 12: Reporting (`wi report`)

Responsibility: Offline Intelligence

1. `wi report generate [TABLE] --out [FILE]` - **Sweetviz:** Generate full HTML EDA dashboard.

2. `wi report diff [TABLE] --snapshot [ID]` - Generate "Before/After" comparison HTML.
3. `wi report validation [TABLE]` - Generate Data Quality Report HTML.
4. `wi report schema-doc` - Generate static Data Dictionary website.
5. `wi report audit-timeline` - Generate visual timeline of data changes.
6. `wi report map [TABLE]` - Generate offline Leaflet.js HTML map.
7. `wi report summary [TABLE]` - Generate one-page Executive Summary PDF.
8. `wi report dependencies` - Generate DAG visualization of table lineage.
9. `wi report profile [TABLE]` - Dump raw statistical profile JSON.
10. `wi report serve` - Spool up temporary `localhost` server to view reports.

---

3. The "Command Center" (Module 13: Interactive TUI)

Accessible via `wi interact`, this module launches a full-screen application.

- **Widget 1: The Grid.** A scrolling, sortable, filtering spreadsheet view of the data.
- **Widget 2: The Monitor.** Real-time CPU/Ram usage and background Ops job status.
- **Widget 3: The Builder.** A visual tool to construct `wi query` strings using dropdowns and sliders.

---

4. Developer Guidelines

- **No Cloud Imports:** Do not use `requests`, `boto3`, or `openai`.
- **Lazy Loading:** Libraries like `pandas` and `scikit-learn` are heavy. Import them *inside* the function scope, not at the top level, to keep CLI startup time under 200ms.
- **Error Handling:** Use `rich.console` for all user feedback. Never print raw tracebacks to the user; log them to `~/.wi/error.log`.