Wrought Iron Commands

To compete with enterprise tools (or Python scripts written by senior engineers), every single command must expose the full power of the underlying library (`pandas`, `scikit-learn`, `cryptography`, etc.) through CLI flags.

We call this the **"Surface-to-Core" API philosophy**: The default usage is simple (`wi clean ml-impute users birth_year`), but the *optional* flags expose the full Core API of the Python library.

Here are 5 concrete examples showing this "Deep Argument" structure applied to different modules, proving that this depth extends far beyond just machine learning.

---

1. The Geo Module: Deep Clustering

Instead of just "grouping points," we expose the full **DBSCAN** algorithm parameters from `scikit-learn`.

```
wi geo cluster users \
  --eps 0.5 \               # (DBSCAN) Max distance between points (km)
  --min-samples 10 \        # (DBSCAN) Min points to form a cluster
  --metric haversine \      # (DBSCAN) Use earth-curvature math, not Euclidean
  --algorithm ball_tree \   # (Optimization) Faster spatial indexing method
  --n-jobs -1 \             # (Performance) Use all CPU cores
  --out-col catchment_zone \# (Save) Name of the new column
  --noise-label "Rural" \   # (Labeling) What to call unclustered points
  --dry-run                 # Preview cluster sizes without saving
```

2. The Aggregate Module: Deep Statistics

Instead of just "grouping," we expose the full power of Pandas `groupby()` and `agg()`.

```
wi aggregate groupby transactions amount \
  --by suburb,category \    # Multi-level grouping
  --agg mean,sum,std,max \  # Multiple statistical operations
  --sort-by sum \           # Order result by total amount
  --ascending False \       # Highest first
  --min-count 5 \           # Drop groups with <5 records (Privacy)
  --dropna \                # Exclude groups where 'suburb' is NULL
  --format markdown \       # Output format (Table, CSV, Markdown, JSON)
  --out report_Q3.md        # Save directly to file
```

3. The Query Module: Deep Filtering

Exposing the granularity of Pandas querying and string handling.

```
wi query search students "Smith" \
  --col name,parent_name \  # Limit search scope
  --case-sensitive False \  # Ignore capitalization
  --regex \                 # Treat "Smith" as Regex pattern
  --fuzzy-threshold 85 \    # Allow slight misspellings ("Smyth")
  --context 50 \            # Show 50 chars of text around the match
  --limit 100 \             # Stop after 100 hits
  --output-mode id-only     # Only return IDs for piping to another command
```

4. The Ops Module: Deep Scheduling

Exposing system-level cron/task controls.

```
wi ops schedule create "clean-nightly" \
  --cmd "wi clean impute-group users" \
  --cron "0 3 * * *" \      # Run at 3:00 AM daily
  --timeout 3600 \          # Kill if runs longer than 1 hour
  --retry 3 \               # Retry 3 times if it fails
  --on-fail-email "admin@secure.mil" \ # Local alert hook
  --cpu-limit 50% \         # Nice value (don't hog the server)
  --log-level DEBUG         # Verbose logging for debugging
```

5. The Audit Module: Deep Forensics

Exposing the cryptographic controls.

```
wi audit hash-verify users \
  --algorithm sha256 \      # Hashing standard
  --exclude-cols updated_at \# Ignore dynamic timestamp columns
  --salt "random-seed" \    # Salt the hash for extra security
  --strict \                # Fail if schema changed (not just data)
  --chunk-size 10000 \      # Optimize memory for massive tables
  --report-format pdf \     # Generate formal certificate
  --signer-key private.pem  # Digitally sign the verification report
```

The "Help" Experience

Because these commands are deep, the Developer Experience (DX) must be perfect. Using `Typer`, typing `--help` for any command will auto-generate this documentation in the terminal:

```
$ wi clean ml-impute --help

Usage: wi clean ml-impute [OPTIONS] TABLE COL

  Impute missing values using K-Nearest Neighbors (KNN).

Arguments:
  TABLE  Target database table  [required]
  COL    Column with missing values  [required]

Options:
  --method [knn|iterative]  Imputation strategy [default: knn]
  --neighbors INT           Number of neighbors (k) [default: 5]
  --weights [uniform|distance]
                            Weight function used in prediction [default: uniform]
  ...
```

This confirms that **Wrought Iron** is not just a wrapper; it is a full-featured interface to the Python Data Science stack.