Table of Contents

SEIR ETL

ETL runners for SEIR data warehouse jobs. This repo provides a small Python 3 service that connects to OLTP (read) and OLAP (write) MySQL databases, discovers which ETL "processor" to run from configuration in the OLAP DB, and then executes either a daily incremental load or a full refresh. Containers for individual ETLs are defined via docker-compose.

Overview

- Language: Python 3.9 (Docker image: python:3.9-slim)
- Package manager: pip with requirements.txt
- Primary dependency: PyMySQL
- Entry point: main.py
- Processors (examples in this repo):
    - etl_processors.hpv.Processor — HPV vaccinations facts
    - etl_processors.child_performance_indicators.Processor — Child vaccine performance indicators summary
- Logging: to logs/etl_<ETL_NAME>.log (directory configurable via LOG_DIR)
- Time zone (container): Asia/Karachi (set in Dockerfile)

The service loads configuration for a given ETL_NAME from table etl_types in the OLAP database (see Configuration section) and then orchestrates the run:

- Acquire a DB lock to ensure single concurrency per ETL
- Check dependency (if defined in config)
- Determine mode (incremental vs refresh)
- Run the configured processor
- Record job status into etl_jobs

Requirements

- Python 3.9+
- MySQL or MariaDB servers for:
    - OLTP (read-heavy source)
    - OLAP (DWH write target and ETL metadata tables)
- Network access from runner to both databases

Python packages (from requirements.txt):

- PyMySQL

Project Structure

```
├── Dockerfile
├── docker-compose.yml
├── requirements.txt
├── main.py                # Application entry point
├── etl.py                 # Legacy script – keep for reference (not used by docker entrypoint)
├── core/
│   ├── __init__.py
│   ├── etl_processor_base.py  # Base class + loader for processors
│   ├── etl_status.py          # Enum-like statuses used by runs
│   └── etl_utils.py           # DB config loading, jobs table helpers, locks, etc.
├── etl_processors/
│   ├── __init__.py
│   ├── hpv.py
│   └── child_performance_indicators.py
│   └── women_performance_indicators.py
└── logs/
    └── etl_HPV_VACCINATIONS.log (example log file)
```

Configuration

Configuration is provided via environment variables and metadata tables in the OLAP database.

Environment variables (used by main.py):

- DB_OLTP_HOST: Hostname/IP of OLTP (read) database
- DB_OLTP_USER: Username for OLTP
- DB_OLTP_PASSWORD: Password for OLTP
- DB_OLTP_NAME: Database/schema name for OLTP
- DB_OLAP_HOST: Hostname/IP of OLAP (write + metadata) database
- DB_OLAP_USER: Username for OLAP
- DB_OLAP_PASSWORD: Password for OLAP
- DB_OLAP_NAME: Database/schema name for OLAP
- GET_LOCK_TIMEOUT: Seconds to wait on named lock (default: 5)
- LOG_DIR: Directory for logs (default: /logs)
- ETL_NAME: Name of the ETL to run (must exist in metadata table)

Metadata tables expected in OLAP database (used by core/etl_utils.py):

- etl_types: stores ETL configuration. Expected columns returned by query in load_etl_config():
    - etl_id

    - etl_name
    - depends_on_etl_id
    - processor_module (e.g., etl_processors.hpv)
    - full_refresh_start_date
    - batch_size
    - extra_config (JSON or NULL)
    - behaviour (ENUM: INCREMENTAL | REFRESH)
- etl_jobs: stores job runs (id, etl_id, created_at, status, last_run, notes)

Note: The schema DDL for these tables is not included in this repo. See TODOs.

Git Branches:

ETL folder: /opt/apps/etl-zm
Repo: https://github.com/CHS-The-Aga-Khan-University/seir-etl
On Production branch: main
On Staging branch: dev

Running

1) Local (Python)

Create a virtual environment and install dependencies:

```
python3.9 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Export required environment variables and run:

```
export DB_OLTP_HOST=<oltp-host>
export DB_OLTP_USER=<oltp-user>
export DB_OLTP_PASSWORD=<oltp-pass>
export DB_OLTP_NAME=<oltp-db>
export DB_OLAP_HOST=<olap-host>
export DB_OLAP_USER=<olap-user>
export DB_OLAP_PASSWORD=<olap-pass>
export DB_OLAP_NAME=<olap-db>
export LOG_DIR=./logs
export ETL_NAME=HPV_VACCINATIONS  # or PERFORMANCE_INDICATORS,
etc.
```

```
python main.py
```

Logs will be written to logs/etl_<ETL_NAME>.log.

2) Docker

Build the image:

```
docker build -t zm-etl:latest .
```

Run a container (example):

```
docker run --rm \
  -e DB_OLTP_HOST=<oltp-host> \
  -e DB_OLTP_USER=<oltp-user> \
  -e DB_OLTP_PASSWORD=<oltp-pass> \
  -e DB_OLTP_NAME=<oltp-db> \
  -e DB_OLAP_HOST=<olap-host> \
  -e DB_OLAP_USER=<olap-user> \
  -e DB_OLAP_PASSWORD=<olap-pass> \
  -e DB_OLAP_NAME=<olap-db> \
  -e ETL_NAME=HPV_VACCINATIONS \
  -e GET_LOCK_TIMEOUT=5 \
  -e LOG_DIR=/logs \
  -v $(pwd)/logs:/logs \
  zm-etl:latest
```

The container ENTRYPOINT runs:

```
python main.py
```

3) Docker Compose

This repo includes a docker-compose.yml that defines two services as examples, sharing common env and volume mounts:

- etl-hpv with ETL_NAME=HPV_VACCINATIONS
- etl-performance-indicators with ETL_NAME=PERFORMANCE_INDICATORS

Before running an ETL, build the Docker images:

```
docker compose build
```

Run the required ETL service:

```
docker compose run --rm etl-hpv
```

Replace etl-hpv with the appropriate service name to execute a different ETL. The --rm flag automatically removes the container once the ETL execution completes.

Logs for each service are written under ./logs on the host.

Security note: The sample compose file currently contains concrete credentials/hosts for a specific network. Replace these with environment overrides or .env values before using outside the intended environment.

Processors

Processors live in etl_processors/ and subclass core.ETLProcessorBase. Each processor must implement:

- get_affected_dates(last_run_ts) — returns list of affected dates (YYYY-MM-DD) or None to skip
- process_daily(affected_dates) — recompute facts for each date in a single transaction
- full_refresh() — rebuild facts since a configured start date in a single transaction

Examples in this repo:

- etl_processors.hpv.Processor: reads from unfepi.* OLTP tables and writes unfepi_dwh.hpv_facts.
- etl_processors.child_performance_indicators.Processor: aggregates from unfepi_dwh.dmp_child_vaccine_incentive_reminder (note: code streams from OLTP cursor) into unfepi_dwh.dmp_summary_child_vac.
- etl_processors.women_performance_indicators.Processor: aggregates from unfepi_dwh.dmp_women_vaccine (note: code streams from OLTP cursor) into unfepi_dwh.dmp_summary_women_vac.

Actual table names and schemas must exist in your OLTP/OLAP instances.

Scripts and Tasks

- Entry point: main.py
- Dockerfile ENTRYPOINT: python main.py
- docker-compose services: see docker-compose.yml
- Legacy script: etl.py (older, monolithic implementation). Prefer main.py + processor classes. TODO: evaluate removal or document intended usage if still needed.

Logs

- Log directory defaults to /logs (configurable with LOG_DIR). In docker-compose, this is mounted to ./logs on the host.
- Per-ETL log file: etl_<ETL_NAME>.log.

Tests

No automated tests are included in this repository.

TODOs:

- Add unit tests for processor orchestration and utility functions
- Add integration tests (requires test MySQL instances and seed data)

Troubleshooting

- "Another instance is running. Exiting." — A named lock could not be acquired. Wait until other run finishes or check for orphaned locks in OLAP DB.
- "Dependency not satisfied. Exiting." — The configured depends_on_etl_id has no successful etl_jobs yet. Ensure the dependency ETL runs successfully first.
- Database errors — Check connectivity, credentials, and that required tables (etl_types, etl_jobs, and processor source/target tables) exist.
- No affected dates — For incremental runs, get_affected_dates can return none; the job will be marked SKIPPED.

License

No license file was found in this repository.

Script Details:

Child Defaulter ETL

Script Name: child_defaulter.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_ChildDefaulter

Description

The Child Defaulter ETL populates the unfepi_dwh.dmp_child_defaulter fact table, which identifies children who have defaulted on scheduled vaccinations. A child is considered a defaulter when a vaccine remains unadministered for more than 28 days after its scheduled due date.
The ETL supports two execution modes:

- Full Refresh – Rebuilds the entire table from the source data.
- Incremental Refresh – Rebuilds records only for children affected since the last successful ETL execution.

This processor is intended to minimize daily processing while still allowing a complete rebuild whenever business logic changes or historical corrections are required.

Data Sources

The ETL reads data from the following tables/views:

Object	Purpose
unfepi_dwh.dmp_child_biography	Child demographic and enrollment information.
unfepi_dwh.dmp_child_vaccine_incentive_reminder	Vaccination history, due dates, vaccination status, centers, and vaccinators.
unfepi_dwh.vaccine_gap_v	Vaccine schedule and recommended age gaps between vaccines.
unfepi_dwh.vaccine_prereq_gap_v	Vaccine prerequisite mapping.
unfepi.childdailyaudit	Detects children whose vaccination data has changed since the previous ETL execution.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilds of the warehouse table.

Process

1. Truncate unfepi_dwh.dmp_child_defaulter.
2. Execute a single INSERT INTO ... SELECT statement.
3. Populate the table directly from the warehouse source tables.
4. Commit the transaction.

Unlike the incremental process, the full refresh performs all aggregation within MySQL, avoiding Python-side row processing and significantly reducing application overhead.

Incremental Refresh

The incremental process rebuilds only affected children.

Step 1 – Identify affected children

Affected child IDs are collected from three sources:

- Vaccination reminder records created after the previous successful ETL run.
- Entries in childdailyaudit modified after the previous successful run.
- Children whose vaccination due dates have entered the 28-day default window since the last execution.

Step 2 – Process affected children

For each batch of child IDs:

1. Delete existing records for those children from dmp_child_defaulter.
2. Recalculate defaulter information.
3. Insert the newly generated records.
4. Commit once all batches complete successfully.

This approach ensures existing facts remain synchronized even when vaccination history changes.

Defaulter Identification Logic

For every enrolled child, the ETL evaluates each vaccine in the routine immunization schedule. A child is considered a defaulter when:

- The vaccine is due.
- The child is at least 28 days beyond the recommended vaccination gap.
- The vaccine due date is on or after 1 January 2019.
- The vaccination has either:

    - not been administered, or
    - was administered more than 28 days after the scheduled due date.

The ETL also derives:

- Enrollment age
- Age at default
- Defaulted vaccine
- Enrollment center
- Defaulting center
- Enrollment vaccinator
- Defaulting vaccinator
- Vaccination event type
- Geographic ancestry

Output

The ETL populates:
unfepi_dwh.dmp_child_defaulter

Each record represents a child–vaccine combination where the child has defaulted according to the business rules.
Key output attributes include:

- Child identifier
- Enrollment age
- Default age
- Enrollment vaccine
- Defaulted vaccine
- Vaccination due date
- Default date
- Vaccination status
- Vaccination date
- Enrollment center
- Default center
- Enrollment vaccinator
- Default vaccinator
- Geographic ancestry

Error Handling

The processor executes all database modifications within a transaction.

Full Refresh

- The destination table is truncated before rebuilding.
- A single INSERT ... SELECT statement populates the table.

- If the insert fails, the transaction is rolled back where applicable. (Note that in MySQL, TRUNCATE TABLE causes an implicit commit, so the table remains empty if the subsequent insert fails.)

Incremental Refresh

- Child records are processed in configurable batches.
- If any batch fails, the transaction is rolled back.
- Because the ETL records execution status, the same affected child IDs are picked up again during the next successful run.

Deployment Fallback:

If the initial full-refresh deployment fails after TRUNCATE TABLE has executed, run the existing ‘unfepi_dwh.DMP_ChildDefaulter’ stored procedure to restore the contents of dmp_child_defaulter before retrying the ETL deployment. Since TRUNCATE auto-commits, the table cannot be restored through a transaction rollback.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding the warehouse after business rule changes.
- Recovering from data inconsistencies.

The processor truncates the destination table and reconstructs it entirely from the warehouse source tables.

Incremental Refresh

Used during normal scheduled ETL execution.
Only children affected since the previous successful run are recalculated, significantly reducing processing time while keeping the warehouse synchronized with operational data.

Child Performance Indicators ETL

Script Name: child_performance_indicators.py
Current Deployment: Production (main branch)
Full Refresh Stored Procedure: N/A
Process: New ETL for populating and incrementally updating summary tables

Description

The Child Vaccination Summary ETL populates the warehouse summary tables that provide aggregated vaccination statistics by vaccination date, center, vaccinator, and event type. The processor produces daily summary data used by dashboards and operational reporting.
The ETL currently populates the following warehouse tables:

- unfepi_dwh.dmp_summary_child_vac
- unfepi_dwh.dmp_summary_visits_by_vaccinator

The ETL supports two execution modes:

- Full Refresh – Rebuilds all summary records from the configured historical start date through the current date.
- Incremental Refresh – Rebuilds summary records only for vaccination dates affected since the previous successful ETL execution.

This processor minimizes daily processing while allowing complete historical rebuilding whenever business rules change or historical vaccination data is corrected.

Data Sources

The ETL reads data from the following tables:

Object	Purpose
unfepi_dwh.dmp_child_vaccine_incentive_reminder	Source of vaccination records, vaccination status, vaccination event type, centers, vaccinators, and encounter information.
unfepi_dwh.dmp_child_biography	Child enrollment information used to calculate enrollment counts.
unfepi.contactnumber	Determines whether enrolled or follow-up children have registered primary mobile numbers.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of historical summary tables.

Process

1. Determine the refresh period from full_refresh_start_date through the current date.
2. Delete existing records from dmp_summary_child_vac for the refresh period.
3. Recalculate and populate daily vaccination summaries for every date in the refresh range.
4. Delete existing records from dmp_summary_visits_by_vaccinator for the refresh period.
5. Recalculate and populate daily vaccinator visit summaries for every date in the refresh range.
6. Commit each successfully processed day.

The processor rebuilds historical summaries one day at a time, reducing memory consumption and allowing long historical periods to be processed efficiently.

Incremental Refresh

The incremental process rebuilds only dates whose vaccination records have changed.

Step 1 – Identify affected dates

Affected vaccination dates are determined from:

- Vaccination records created after the previous successful ETL execution.
- Only records with vaccinationStatus = 'VACCINATED' are considered.

Step 2 – Process affected dates

For each affected vaccination date:

- Delete existing summary records from dmp_summary_child_vac.
- Recalculate vaccination summary statistics.
- Delete existing summary records from dmp_summary_visits_by_vaccinator.
- Recalculate vaccinator visit summary statistics.
- Commit once all affected dates have been processed successfully.

This approach ensures summary tables remain synchronized whenever vaccination records are inserted or updated.

Aggregation Logic

The ETL derives daily summary metrics from completed vaccination records.

For dmp_summary_child_vac, the processor aggregates:

- Vaccination center

- Vaccinator
- Vaccination date
- Vaccine
- Vaccination event type (Fixed or Outreach)
- Number of enrollments
- Number of immunizations

Vaccination event types are normalized into two reporting categories:
- FIXED
- OUTREACH

For dmp_summary_visits_by_vaccinator, the processor aggregates:
- Vaccinator
- Vaccination date
- Event type
- Enrollment visits
- Follow-up visits
- Enrollment mobile registrations
- Follow-up mobile registrations

Follow-up statistics are calculated using vaccination encounter types, while mobile registration counts are derived from active primary contact numbers linked to vaccinated children.

Output

The ETL populates:
- unfepi_dwh.dmp_summary_child_vac
- unfepi_dwh.dmp_summary_visits_by_vaccinator

dmp_summary_child_vac

Each record represents vaccination activity for a unique combination of:
- Vaccination date
- Vaccination center
- Vaccinator
- Vaccine
- Event type

Key output attributes include:
- Center ID
- Vaccinator ID
- Vaccination date
- Vaccine ID
- Event type
- Enrollment count
- Immunization count

dmp_summary_visits_by_vaccinator

Each record represents daily vaccinator activity grouped by:

- Vaccination date
- Vaccinator
- Event type

Key output attributes include:

- Vaccinator ID
- Vaccination date
- Event type
- Enrollment count
- Follow-up count
- Mobile enrollment count
- Mobile follow-up count

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- Existing records for the configured refresh period are deleted before rebuilding.
- Each processed day is committed independently.
- If processing fails before a day's commit, the transaction for that day is rolled back.
- Previously committed dates remain intact and do not require rebuilding.

Incremental Refresh

- Summary records are rebuilt only for affected vaccination dates.
- If processing fails before commit, all uncommitted changes are rolled back.
- Because the ETL records execution status, the same affected vaccination dates are processed again during the next successful run.

Deployment Note:

Since this is a new ETL process and no previous stored procedure exists for rebuilding these summary tables, the initial full refresh should be validated in staging before production deployment.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.

- Rebuilding historical summary tables.
- Applying business rule changes.
- Recovering from historical data inconsistencies.

The processor rebuilds all vaccination summaries from the configured historical start date through the current date.

Incremental Refresh

Used during normal scheduled ETL execution.
Only vaccination dates affected since the previous successful ETL run are recalculated, significantly reducing processing time while keeping the warehouse summary tables synchronized with operational vaccination data.

Child Vaccination Aggregate ETL

Script Name: child_vaccination_aggregate.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_ChildVaccinationAggregate

Description

The Child Vaccination Aggregate ETL populates the unfepi_dwh.dmp_child_vaccination_aggregate summary table, which provides aggregated vaccination statistics by vaccination date, center, vaccine, vaccination age, and vaccination event type.
The ETL supports two execution modes:

- Full Refresh – Rebuilds aggregate records from the configured full refresh start date through the current date.
- Incremental Refresh – Rebuilds aggregate records only for vaccination dates affected since the last successful ETL execution.

This processor minimizes daily processing by recalculating only affected dates while allowing complete historical rebuilding when required.

Data Sources

The ETL reads data from the following tables/views:

Object	Purpose
unfepi_dwh.dmp_child_vaccine_incentive_reminder	Vaccination records, vaccination dates, vaccine details, vaccination status, centers, and event types.
unfepi_dwh.dmp_child_biography	Child demographic information including gender and UC details.
unfepi_dwh.dmp_child_defaulter	Identifies defaulters covered during vaccination events.
unfepi_dwh.location_hierarchy_ancester	Provides UC hierarchy mapping for location analysis.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of the aggregate table.

Process

- Delete existing records from unfepi_dwh.dmp_child_vaccination_aggregate from the configured refresh start date.
- Generate daily aggregate records for each date from the refresh start date through the current date.
- Insert calculated aggregate records.
- Commit each successfully processed day.

The processor rebuilds data one day at a time to reduce memory usage and support large historical refreshes.

Incremental Refresh

The incremental process rebuilds only affected vaccination dates.

Step 1 – Identify affected dates

Affected vaccination dates are collected from:

- Vaccination records created after the previous successful ETL execution.
- Only records with vaccinationstatus = 'VACCINATED' are considered.

Step 2 – Process affected dates

For each affected vaccination date:

- Delete existing records for that date from dmp_child_vaccination_aggregate.
- Recalculate vaccination aggregate information.
- Insert newly generated aggregate records.
- Commit once all affected dates are processed successfully.

This approach ensures aggregate data remains synchronized when vaccination records are added or updated.

Aggregation Logic

The ETL aggregates vaccination activity by:

- Vaccination date
- Vaccination center
- UC
- Vaccine
- Vaccination age group
- Vaccination event type

The ETL derives:

- Total vaccinations
- Male vaccinations
- Female vaccinations
- Total children vaccinated

- Male children vaccinated
- Female children vaccinated
- Inside UC vaccinations
- Outside UC vaccinations
- Shifted vaccination counts
- Defaulters covered
- Male defaulters covered
- Female defaulters covered

Output

The ETL populates:
unfepi_dwh.dmp_child_vaccination_aggregate
Each record represents aggregated vaccination activity for a unique combination of vaccination date, vaccine, center, event type, and vaccination age group.
Key output attributes include:

- Vaccination date
- Vaccination center ID
- UC ID
- UC name
- Vaccine ID
- Vaccination age years
- Vaccination event type
- Vaccination counts
- Child counts
- Inside/outside UC counts
- Shifted vaccination counts
- Defaulters covered counts

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- Existing aggregate records are deleted before rebuilding.
- Each processed day is committed independently.
- If processing fails before a day's commit, the transaction for that day is rolled back.
- Previously committed dates remain intact.

Incremental Refresh

- Aggregate records are rebuilt only for affected vaccination dates.
- If processing fails before commit, changes are rolled back.

- Because the ETL records execution status, the same affected vaccination dates are processed again during the next successful run.

Deployment Fallback:

If the initial full-refresh deployment fails after the delete operation has been executed, run the existing unfepi_dwh.DMP_ChildVaccinationAggregate stored procedure to restore the contents of dmp_child_vaccination_aggregate before retrying the ETL deployment.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding historical aggregate data.
- Applying business rule changes.
- Recovering from historical data inconsistencies.

The processor rebuilds aggregate records from the configured refresh start date through the current date.

Incremental Refresh

Used during normal scheduled ETL execution.
Only vaccination dates affected since the previous successful run are recalculated, reducing processing time while keeping the aggregate table synchronized with operational vaccination data.

Child Vaccination Aggregate ETL

Script Name: child_vaccination_aggregate.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_ChildVaccinationAggregate

Description

The Child Vaccination Aggregate ETL populates the unfepi_dwh.dmp_child_vaccination_aggregate summary table, which provides aggregated vaccination statistics by vaccination date, center, vaccine, vaccination age, and vaccination event type.
The ETL supports two execution modes:

- Full Refresh – Rebuilds aggregate records from the configured full refresh start date through the current date.
- Incremental Refresh – Rebuilds aggregate records only for vaccination dates affected since the last successful ETL execution.

This processor minimizes daily processing by recalculating only affected dates while allowing complete historical rebuilding when required.

Data Sources

The ETL reads data from the following tables/views:

Object	Purpose
unfepi_dwh.dmp_child_vaccine_incentive_reminder	Vaccination records, vaccination dates, vaccine details, vaccination status, centers, and event types.
unfepi_dwh.dmp_child_biography	Child demographic information including gender and UC details.
unfepi_dwh.dmp_child_defaulter	Identifies defaulters covered during vaccination events.
unfepi_dwh.location_hierarchy_ancester	Provides UC hierarchy mapping for location analysis.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of the aggregate table.

Process

- Truncate unfepi_dwh.dmp_child_vaccination_aggregate.
- Execute a single INSERT INTO ... SELECT statement.
- Generate daily aggregate records for each date from the refresh start date through the current date.
- Commit the transaction.

The full refresh performs all aggregation within MySQL, avoiding Python-side row processing and reducing application overhead.

Incremental Refresh

The incremental process rebuilds only affected vaccination dates.

Step 1 – Identify affected dates

Affected vaccination dates are collected from:

- Vaccination records created after the previous successful ETL execution.
- Only records with vaccinationstatus = 'VACCINATED' are considered.

Step 2 – Process affected dates

For each affected vaccination date:

- Delete existing records for that date from dmp_child_vaccination_aggregate.
- Recalculate vaccination aggregate information.
- Insert newly generated aggregate records.
- Commit once all affected dates are processed successfully.

This approach ensures aggregate data remains synchronized when vaccination records are added or updated.

Aggregation Logic

The ETL aggregates vaccination activity by:

- Vaccination date
- Vaccination center
- UC
- Vaccine
- Vaccination age group
- Vaccination event type

The ETL derives:

- Total vaccinations
- Male vaccinations
- Female vaccinations
- Total children vaccinated
- Male children vaccinated

- Female children vaccinated
- Inside UC vaccinations
- Outside UC vaccinations
- Shifted vaccination counts
- Defaulters covered
- Male defaulters covered
- Female defaulters covered

Output

The ETL populates: unfepi_dwh.dmp_child_vaccination_aggregate

Each record represents aggregated vaccination activity for a unique combination of vaccination date, vaccine, center, event type, and vaccination age group.

Key output attributes include:

- Vaccination date
- Vaccination center ID
- UC ID
- UC name
- Vaccine ID
- Vaccination age years
- Vaccination event type
- Vaccination counts
- Child counts
- Inside/outside UC counts
- Shifted vaccination counts
- Defaulters covered counts

Error Handling

The processor executes all database modifications within a transaction.

Full Refresh

- The destination table is truncated before rebuilding.
- A single INSERT ... SELECT statement repopulates the table.
- If the insert fails, the transaction is rolled back where applicable. (Note that in MySQL, TRUNCATE TABLE causes an implicit commit, so the table remains empty if the subsequent insert fails.)

Incremental Refresh

- Aggregate records are rebuilt only for affected vaccination dates.
- If processing fails before commit, changes are rolled back.

- Because the ETL records execution status, the same affected vaccination dates are processed again during the next successful run.

Deployment Fallback:

If the initial full-refresh deployment fails after TRUNCATE TABLE has executed, run the existing unfepi_dwh.DMP_ChildVaccinationAggregate stored procedure to restore the contents of dmp_child_vaccination_aggregate before retrying the ETL deployment. Since TRUNCATE auto-commits, the table cannot be restored through a transaction rollback.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding historical aggregate data.
- Applying business rule changes.
- Recovering from historical data inconsistencies.

The processor rebuilds aggregate records from the configured refresh start date through the current date.

Incremental Refresh

Used during normal scheduled ETL execution.
Only vaccination dates affected since the previous successful run are recalculated, reducing processing time while keeping the aggregate table synchronized with operational vaccination data.

Child Vaccination Aggregate Monthly ETL

Script Name: child_vaccination_aggregate_monthly.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_ChildVaccinationAggregateMonthly

Description

The Child Vaccination Aggregate Monthly ETL populates the unfepi_dwh.dmp_child_vaccination_aggregate_monthly summary table, which stores monthly aggregated vaccination statistics derived from the daily vaccination aggregate table.
The ETL supports two execution modes:

- Full Refresh – Rebuilds the entire monthly aggregate table from the configured historical start date.
- Incremental Refresh – Rebuilds only the months affected since the last successful ETL execution.

This processor minimizes daily processing while allowing complete rebuilding whenever historical vaccination aggregates change.

Data Sources

The ETL reads data from the following table:

Object	Purpose
unfepi_dwh.dmp_child_vaccination_aggregate	Daily vaccination aggregate data used to generate monthly summaries.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of the monthly aggregate table.

Process

- Truncate unfepi_dwh.dmp_child_vaccination_aggregate_monthly.
- Execute a single INSERT INTO ... SELECT statement.
- Aggregate daily vaccination summaries into monthly summaries.
- Commit the transaction.

The full refresh performs all aggregation within MySQL, avoiding Python-side row processing and reducing application overhead.

Incremental Refresh

The incremental process rebuilds only affected months.

Step 1 – Identify affected months

Affected year-month combinations are collected from:

- Vaccination records created after the previous successful ETL execution.
- Only records with vaccinationStatus = 'VACCINATED' are considered.

Step 2 – Process affected months

For each affected month:

- Delete existing records for that month from dmp_child_vaccination_aggregate_monthly.
- Recalculate monthly aggregates from dmp_child_vaccination_aggregate.
- Insert the newly generated records.
- Commit once all affected months complete successfully.

This approach ensures monthly summaries remain synchronized whenever daily aggregates change.

Aggregation Logic

The ETL aggregates daily vaccination summaries into monthly totals grouped by:

- Vaccination year
- Vaccination month
- Vaccination center
- UC
- Vaccine
- Vaccination age (years)
- Vaccination event type

The ETL derives:

- Vaccinations
- Vaccinations by gender
- Children vaccinated
- Children by gender
- Defaulters covered
- Defaulters covered by gender
- Inside UC vaccinations
- Outside UC vaccinations
- Inside/Outside UC by gender
- Defaulters covered inside/outside UC
- Defaulters covered inside/outside UC by gender
- Shifted outside UC (Male)
- Shifted outside UC (Female)

Output

The ETL populates: unfepi_dwh.dmp_child_vaccination_aggregate_monthly

Each record represents a unique combination of:

- Vaccination year
- Vaccination month
- Vaccination center
- UC
- Vaccine
- Vaccination age
- Vaccination event type

Key output attributes include:

- Vaccination year
- Vaccination month
- Vaccination center
- UC
- Vaccine
- Vaccination age
- Vaccination event type
- Vaccination counts
- Child counts
- Defaulters covered
- Inside/Outside UC statistics
- Shifted outside UC statistics

Error Handling

The processor executes all database modifications within a transaction.

Full Refresh

- The destination table is truncated before rebuilding.
- A single INSERT ... SELECT statement repopulates the table.
- If the insert fails, the transaction is rolled back where applicable. (Note that in MySQL, TRUNCATE TABLE causes an implicit commit, so the table remains empty if the subsequent insert fails.)

Incremental Refresh

- Monthly records are processed for affected months.
- If processing fails, the transaction is rolled back.
- Because the ETL records execution status, the same affected months are processed again during the next successful run.

Deployment Fallback:

If the initial full-refresh deployment fails after TRUNCATE TABLE has executed, run the existing unfepi_dwh.DMP_ChildVaccinationAggregateMonthly stored procedure to restore the contents of dmp_child_vaccination_aggregate_monthly before retrying the ETL deployment. Since TRUNCATE auto-commits, the table cannot be restored through a transaction rollback.

Master Child Biography Data ETL

Script Name: master_child_data_bio.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_MasterChildData
Process: Part of DMP_MasterChildData SP, needs to run after master_child_data.py

Description

The Child Biography ETL populates the warehouse table unfepi_dwh.dmp_child_biography, which serves as the master child dimension for reporting and downstream ETL processes.
The processor consolidates demographic, enrollment, address, contact, vaccination enrollment, consent, and maternal information for each child into a single denormalized warehouse table.
The ETL supports two execution modes:

- Full Refresh – Rebuilds all child biography records from the configured historical start date through the current date.
- Incremental Refresh – Rebuilds biography records only for children whose information has changed since the previous successful ETL execution.

This approach minimizes daily processing while allowing complete rebuilding whenever historical child information changes.

Data Sources

The ETL reads data from the following tables.

Object	Purpose
unfepi.child	Primary child demographic and enrollment information.
unfepi.childdailyaudit	Identifies children modified since the previous ETL execution.
unfepi.arm	Child study arm information.
unfepi.program	Program associated with the enrollment arm.
unfepi.childanalysis	Maternal education, language, TT vaccination, place of delivery and related analysis data.

unfepi.address	Primary address and geographic information.
unfepi.identifier	Preferred child identifier.
unfepi.contactnumber	Primary and secondary contact numbers.
unfepi.lotterysms	Lottery and reminder consent preferences.
unfepi_dwh.location_hierarchy_ancester	City, Town, UC hierarchy information.
unfepi_dwh.dmp_child_vaccine_incentive_reminder	Enrollment vaccination, vaccinator and vaccination center information.

Full Refresh

The full refresh is intended for initial deployment or complete rebuilding of the Child Biography dimension.

Process

1. Determine the refresh period from full_refresh_start_date through the current date.
2. Delete all existing biography records associated with audited children in the refresh period.
3. Generate the list of dates between the refresh start date and today.
4. For each date:
    - Identify children modified on that date.
    - Recalculate the latest child biography information.
    - Insert the newly generated records into dmp_child_biography.
    - Commit the transaction.
5. Continue until all dates have been processed.

Processing one day at a time reduces memory consumption and supports rebuilding large historical datasets efficiently.

Incremental Refresh

The incremental refresh rebuilds only biography records for children modified since the previous successful ETL execution.

Step 1 – Identify affected dates

Affected dates are determined from:

- unfepi.childdailyaudit

where:

- logTime falls between the previous successful ETL execution time and the current execution time.
- Only children already mapped into dmp_child_biography are considered.

The processor returns the distinct audit dates requiring rebuilding.

Step 2 – Process affected dates

For each affected date:

1. Delete existing biography records for children modified on that date.
2. Recalculate the latest biography information.
3. Insert the newly generated biography records.
4. Continue until all affected dates have been processed.
5. Commit once all dates complete successfully.

This ensures that updates, corrections and deletions in the operational database are reflected in the warehouse.

Transformation Logic

For each affected child, the ETL consolidates information from multiple operational tables into a single warehouse record.
The transformation derives:

Child Information

- Child Identifier
- Internal Mapped ID
- Child Full Name
- Father Full Name
- Gender
- Birth Date
- Current Age (Days)
- Enrollment Age (Days)
- Date Enrolled
- Medical Record Number
- National Identity Number (CNIC)

- Zero Dose status
- Enrollment Status
- Termination Date
- Termination Reason
- Record Created Date
- Record Last Edited Date

Program Information

- Program
- Program ID
- Arm
- Arm ID

Contact Information

- Primary Contact Number
- Secondary Contact Number
- Contact creation dates
- Contact modification dates

Address Information

- Address Line
- Town
- Town ID
- UC
- UC ID
- City
- City ID
- Province
- Province ID
- Landmark
- High Risk Population Flag
- Geographic Ancestry

Enrollment Vaccination Information

- Enrollment EPI Number
- Enrollment Vaccine
- Enrollment Vaccinator
- Enrollment Vaccination Center
- Vaccination Event Type
- Administrative hierarchy for enrollment location

Maternal Information

- Mother Education
- Mother Language
- Mother TT Vaccination Course
- Mother Identifier
- Place of Delivery

Communication Preferences

The processor captures the latest communication preferences including:

- Approved Lottery
- Approved Reminders
- Preference Change Date
- Preference Created Date
- Preference Last Edited Date

Output

The ETL populates:

- unfepi_dwh.dmp_child_biography

Each record represents one child.
Key output attributes include:

- Child ID
- Mapped ID
- Child Name
- Father Name
- Gender
- Birth Date
- Enrollment Date
- Enrollment Status
- Program
- Arm
- Primary Contact
- Secondary Contact
- Address
- UC
- Town
- City
- Province
- Enrollment Vaccination Information
- Vaccination Center
- Vaccinator
- Maternal Information
- Lottery Consent

- Reminder Consent
- Created Timestamp

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- Existing biography records for the configured refresh period are deleted before rebuilding.
- Each processed day is committed independently.
- If processing fails before a day's commit, that day's transaction is rolled back.
- Previously committed days remain intact and do not require rebuilding.

Incremental Refresh

- Biography records are rebuilt only for affected audit dates.
- If processing fails before the final commit, all uncommitted changes are rolled back.
- Because ETL execution status is tracked, the same affected dates are processed again during the next successful execution.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding the Child Biography dimension.
- Applying business rule changes.
- Recovering from historical data inconsistencies.
- Rebuilding downstream warehouse dependencies.

The processor rebuilds child biography records from the configured historical start date through the current date.

Incremental Refresh

Used during scheduled ETL execution.
Only children whose records were modified since the previous successful ETL execution are rebuilt, significantly reducing processing time while keeping the warehouse synchronized with operational child data.

Master Child Data ETL

Script Name: master_child_data.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_MasterChildData
Process: Part of DMP_MasterChildData SP, needs to run before master_child_data_bio.py

Description

The Child Vaccine Incentive Reminder ETL populates the warehouse table unfepi_dwh.dmp_child_vaccine_incentive_reminder, which serves as the primary vaccination fact table for reporting and downstream ETL processes.
The processor consolidates vaccination events with child, vaccinator, vaccination center, incentive, reminder, tracking, and location information into a single denormalized warehouse table.
The ETL supports two execution modes:

- Full Refresh – Rebuilds vaccination records from the configured historical start date through the current date.
- Incremental Refresh – Rebuilds vaccination records only for dates affected since the previous successful ETL execution.

This approach minimizes daily processing while allowing complete rebuilding whenever historical vaccination information changes.

Data Sources

The ETL reads data from the following tables.

Object	Purpose
unfepi.vaccination	Primary vaccination records.
unfepi.child	Child demographic and enrollment information.
unfepi.childdailyaudit	Identifies vaccination records modified since the previous ETL execution.
unfepi.vaccine	Vaccine master information.
unfepi.vaccinator	Vaccinator demographic information.

unfepi.identifier	Preferred identifiers for children and vaccinators.
unfepi.childincentive	Incentive payment information linked to vaccination records.
unfepi.arm	Study arm and program information.
unfepi.lotterysms	Lottery and reminder consent information.
unfepi.remindersms	Vaccine reminder SMS status and delivery dates.
unfepi.tracking	Vaccinator GPS tracking information.
unfepi.vaccination_paracetamol	Indicates whether paracetamol was provided during vaccination.
unfepi_dwh.vcenter_hierarchy	Vaccination center hierarchy and geographic information.

Full Refresh

The full refresh is intended for initial deployment or complete rebuilding of the vaccination fact table.

Process

1. Determine the refresh period from full_refresh_start_date through the current date.
2. Delete existing warehouse records associated with vaccination audit records in the refresh period.
3. Generate the list of dates between the refresh start date and today.
4. For each date:
    - Identify vaccination records modified on that date.
    - Recalculate the vaccination fact records.
    - Insert the newly generated records into dmp_child_vaccine_incentive_reminder.
    - Commit the transaction.
5. Continue until all dates have been processed.

Processing one day at a time minimizes memory consumption and supports efficient rebuilding of large historical datasets.

Incremental Refresh

The incremental refresh rebuilds only vaccination records modified since the previous successful ETL execution.

Step 1 – Identify affected dates

Affected dates are determined from:

- unfepi.childdailyaudit

where:

- tableName = 'Vaccination'
- logTime is greater than or equal to the previous successful ETL execution time.

The processor returns the distinct audit dates requiring rebuilding.

Step 2 – Process affected dates

For each affected date:

1. Delete existing warehouse records corresponding to vaccination records modified on that date.
2. Recalculate the latest vaccination information.
3. Insert the newly generated vaccination records.
4. Continue until all affected dates have been processed.
5. Commit once all dates complete successfully.

This ensures inserts, updates and deletions made in the operational database are reflected in the warehouse.

Transformation Logic

For each affected vaccination record, the ETL combines information from multiple operational tables into a single warehouse record.
The transformation derives:

Vaccination Information

- Vaccination Record Number
- Child ID
- Vaccination Due Date
- Vaccination Date
- Vaccination Status
- Timeliness Status
- Timeliness Factor
- Vaccination Refusal Date

- Reason Vaccine Not Given
- Vaccination Age (Days)
- Vaccination Age (Months)
- Days Past Due Date
- EPI Number
- Vaccination Created Date
- Vaccination Last Edited Date

Encounter Classification

The ETL derives the encounter type as:

- ENROLLMENT – Vaccination occurred on the child's enrollment date.
- FOLLOWUP – Vaccination occurred after the enrollment date.

Vaccination Center Information

- Vaccination Center ID
- Vaccination Center Identifier
- Vaccination Center Name
- UC
- Town
- District
- Division
- Province
- Geographic Ancestry

Vaccinator Information

- Vaccinator ID
- Vaccinator Identifier
- Vaccinator Name

Vaccine Information

- Vaccine ID
- Vaccine Name
- Vaccination Event Type

Incentive Information

- Incentive Record Number
- Incentive Amount
- Incentive Status
- Incentive Date
- Consumption Date
- Transaction Date

Program Information

- Arm
- Program ID

Communication Preferences

The processor captures:

- Approved Lottery
- Approved Reminders

Reminder SMS Information

For each reminder cycle (1–3):

- Reminder Status
- Due Date
- Sent Date

GPS Tracking Information

- Longitude
- Latitude
- GPS Accuracy
- Tracking Time

Additional Information

- First Vaccination Flag
- Immunized by LHW Flag
- Paracetamol Given Flag

Output

The ETL populates:

- unfepi_dwh.dmp_child_vaccine_incentive_reminder

Each record represents a vaccination event for a child.
Key output attributes include:

- Vaccination Record Number
- Child ID
- Vaccination Date
- Vaccination Status
- Encounter Type
- Vaccine
- Vaccination Center
- Vaccinator
- Vaccination Event Type

- Vaccination Age
- Incentive Information
- Reminder Statuses
- GPS Tracking Information
- Program Information
- Lottery and Reminder Consent
- Created Timestamp

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- Existing warehouse records for the configured refresh period are deleted before rebuilding.
- Each processed day is committed independently.
- If processing fails before a day's commit, that day's transaction is rolled back.
- Previously committed days remain intact and do not require rebuilding.

Incremental Refresh

- Warehouse records are rebuilt only for affected vaccination audit dates.
- If processing fails before the final commit, all uncommitted changes are rolled back.
- Because ETL execution status is tracked, the same affected dates are processed again during the next successful execution.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding the vaccination fact table.
- Applying business rule changes.
- Recovering from historical data inconsistencies.
- Rebuilding downstream warehouse tables that depend on vaccination data.

The processor rebuilds vaccination records from the configured historical start date through the current date.

Incremental Refresh

Used during normal scheduled ETL execution.

Only vaccination records modified since the previous successful ETL execution are recalculated, significantly reducing processing time while keeping the warehouse synchronized with operational vaccination data.

Master Women Data Biography ETL

Script Name: master_women_data_bio.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_MasterWomenData
Process: Should be executed after master_women_data.py

Description

The Women Biography ETL populates the warehouse table unfepi_dwh.dmp_women_biography, which serves as the master dimension for women enrolled in the programme.
The processor consolidates demographic, enrollment, pregnancy, birth registry, contact, address, vaccination enrollment, and antenatal information into a single denormalized warehouse table used by reporting and downstream ETL processes.
The ETL supports two execution modes:

- Full Refresh – Rebuilds all women biography records from the configured historical start date through the current date.
- Incremental Refresh – Rebuilds biography records only for women whose information has changed since the previous successful ETL execution.

To improve performance, the processor creates a temporary aggregated pregnancy analysis table that is reused throughout the execution.

Data Sources

The ETL reads data from the following tables.

Object	Purpose
unfepi.women	Primary demographic and enrollment information for women.
unfepi.womendailyaudit	Identifies women modified since the previous ETL execution.
unfepi.address	Primary address information.
unfepi.contactnumber	Primary and secondary contact numbers.
unfepi.identifier	Preferred woman identifier.

unfepi.user	User who enrolled the woman.
unfepi.encounter	Enrollment encounter information.
unfepi.womenanalysis	Pregnancy, antenatal, birth registry, and delivery information.
unfepi.location	City and birth registry vaccination center information.
unfepi.location_hierarchy_ancester	Geographic hierarchy used to derive ancestry.
unfepi_dwh.dmp_women_vaccine	Enrollment vaccination information.

Full Refresh

The full refresh is intended for initial deployment or complete rebuilding of the Women Biography dimension.

Process

1. Determine the refresh period from full_refresh_start_date through the current date.
2. Delete existing biography records associated with women modified during the refresh period.
3. Create a temporary aggregated women analysis table from womenanalysis.
4. Generate the list of dates between the refresh start date and today.
5. For each date:
    - Identify women modified on that date.
    - Recalculate the latest biography information.
    - Insert the newly generated records into dmp_women_biography.
    - Commit the transaction.
6. Continue until all dates have been processed.

The processor rebuilds data one day at a time, reducing memory consumption while efficiently processing large historical datasets.

Incremental Refresh

The incremental refresh rebuilds only biography records for women modified since the previous successful ETL execution.

Step 1 – Identify affected dates

Affected dates are determined from:

- unfepi.womendailyaudit

where:

- LogTime falls between the previous successful ETL execution time and the current execution time.
- Only women already present in dmp_women_biography are considered.

The processor returns the distinct audit dates requiring rebuilding.

Step 2 – Process affected dates

Before processing any dates, the processor creates a temporary aggregated women analysis table.
For each affected date:

1. Delete existing biography records for women modified on that date.
2. Recalculate the latest biography information.
3. Insert the newly generated biography records.
4. Continue until all affected dates have been processed.
5. Commit once all dates complete successfully.

This ensures inserts, updates and deletions made in the operational database are reflected in the warehouse.

Transformation Logic

For each affected woman, the ETL consolidates information from multiple operational tables into a single warehouse record.
The transformation derives:

Personal Information

- Mapped ID
- Preferred Identifier
- First Name
- Last Name
- Birth Date
- Father's Name
- Husband's Name
- Marital Status
- National Identity Number (CNIC)

- Enrollment Status
- Date Enrolled
- Termination Date
- Termination Reason
- Record Created Date
- Record Last Edited Date

Enrollment Vaccination Information

- Enrollment EPI Number
- Enrollment Vaccine
- Enrollment Vaccinator
- Enrollment Vaccination Center
- Vaccination Event Type
- Enrollment UC
- Enrollment Town
- Enrollment District
- Enrollment Division
- Vaccination Center Hierarchy

Enrollment Information

- Enrolled By (Username)
- Registry Enrollment Flag

Contact Information

- Primary Contact Number
- Secondary Contact Number
- Contact Created Date
- Contact Last Edited Date

Address Information

- Address Line
- Town
- Town ID
- UC
- UC ID
- City
- City ID
- Province
- Province ID
- Landmark
- Geographic Ancestry
- Vaccination Center Ancestry

Pregnancy Information

Current pregnancy details include:

- Antenatal Visit Number
- Pregnancy Duration
- Pregnancy Status
- Next Visit Due Date
- Delivery Due Date
- Delivery Date
- Pregnancy Termination Date
- Live Birth
- Registered Patient Number

Birth Registry Information

- Birth Registry Enrollment Center
- Birth Registry Enrollment Center Name
- Birth Registry Vaccinator
- Birth Registry Enrollment Visit Date

Aggregated Pregnancy Information

Using the temporary aggregation table, the ETL also stores the latest available values for:

- Maximum Live Birth
- Latest Delivery Date
- Latest Registered Patient Number
- Latest Visit Date
- Maximum Antenatal Visit Number
- Latest Delivery Due Date
- Maximum Pregnancy Duration

Temporary Aggregation

To improve performance, the ETL creates a temporary table:
unfepi_dwh.temp_women_analysis_aggregate
The table stores one aggregated record per woman containing the latest values for key pregnancy and delivery attributes.
An index is created on womenId to optimize joins during biography generation.
The temporary table exists only for the duration of the ETL execution.

Output

The ETL populates:

- unfepi_dwh.dmp_women_biography

Each record represents one enrolled woman.

Key output attributes include:

- Woman Identifier
- Mapped ID
- Personal Information
- Enrollment Information
- Vaccination Enrollment Details
- Contact Information
- Address Information
- Pregnancy Information
- Birth Registry Information
- Aggregated Pregnancy Statistics
- Created Timestamp

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- Existing biography records for the configured refresh period are deleted before rebuilding.
- The temporary aggregation table is recreated before processing begins.
- Each processed day is committed independently.
- If processing fails before a day's commit, that day's transaction is rolled back.
- Previously committed days remain intact and do not require rebuilding.

Incremental Refresh

- The temporary aggregation table is recreated at the beginning of each execution.
- Biography records are rebuilt only for affected audit dates.
- If processing fails before the final commit, all uncommitted changes are rolled back.
- Because ETL execution status is tracked, the same affected dates are processed again during the next successful execution.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding the Women Biography dimension.
- Applying business rule changes.
- Recovering from historical data inconsistencies.
- Rebuilding downstream warehouse tables dependent on women biography information.

The processor rebuilds women biography records from the configured historical start date through the current date.

Incremental Refresh

Used during normal scheduled ETL execution.
Only women whose records have been modified since the previous successful ETL execution are rebuilt, significantly reducing processing time while keeping the warehouse synchronized with operational women enrollment and pregnancy data.

Master Women Data ETL

Script Name: master_women_data.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_MasterWomenData
Process: Part of unfepi_dwh.DMP_MasterWomenData, should be executed before master_women_bio.py

Description

The Women Vaccination ETL populates the warehouse table unfepi_dwh.dmp_women_vaccine, which serves as the primary vaccination fact table for women enrolled in the programme.
The processor consolidates vaccination events with vaccine, vaccinator, vaccination centre, geographic hierarchy, tracking, and enrollment information into a single denormalized warehouse table used by reporting and downstream ETL processes.
The ETL supports two execution modes:

- Full Refresh – Rebuilds vaccination records from the configured historical start date through the current date.
- Incremental Refresh – Rebuilds vaccination records only for dates affected since the previous successful ETL execution.

This approach minimizes daily processing while allowing complete rebuilding whenever historical vaccination information changes.

Data Sources

The ETL reads data from the following tables.

Object	Purpose
unfepi.womenvaccination	Primary women vaccination records.
unfepi.women	Women enrollment information including enrollment vaccine.
unfepi.womendailyaudit	Identifies vaccination records modified since the previous ETL execution.
unfepi.vaccine	Vaccine master information.
unfepi.vaccinator	Vaccinator demographic information.

unfepi.identifier	Preferred identifiers for vaccinators and women.
unfepi.location	Vaccination centre identifier information.
unfepi.location_hierarchy_ancester	Geographic hierarchy for vaccination centres (UC, Town, District, Division).
unfepi.tracking	Vaccinator GPS tracking information captured on the vaccination date.

Full Refresh

The full refresh is intended for initial deployment or complete rebuilding of the women vaccination fact table.

Process

1. Determine the refresh period from full_refresh_start_date through the current date.
2. Delete existing warehouse records associated with vaccination audit records in the refresh period.
3. Generate the list of dates between the refresh start date and today.
4. For each date:
    - Identify vaccination records modified on that date.
    - Recalculate the vaccination fact records.
    - Insert the newly generated records into dmp_women_vaccine.
    - Commit the transaction.
5. Continue until all dates have been processed.

Processing one day at a time minimizes memory consumption and supports efficient rebuilding of large historical datasets.

Incremental Refresh

The incremental refresh rebuilds only vaccination records modified since the previous successful ETL execution.

Step 1 – Identify affected dates

Affected dates are determined from:

- unfepi.womendailyaudit

where:

- tableName = 'womenvaccination'
- logTime is greater than or equal to the previous successful ETL execution time.

The processor returns the distinct audit dates requiring rebuilding.

Step 2 – Process affected dates

For each affected date:

1. Delete existing warehouse records corresponding to vaccination records modified on that date.
2. Recalculate the latest vaccination information.
3. Insert the newly generated vaccination records.
4. Continue until all affected dates have been processed.
5. Commit once all dates complete successfully.

This ensures inserts, updates and deletions made in the operational database are reflected in the warehouse.

Transformation Logic

For each affected vaccination record, the ETL combines information from multiple operational tables into a single warehouse record.
The transformation derives:

Vaccination Information

- Women ID
- Vaccination Record Number
- Vaccination Date
- Vaccination Due Date
- Vaccination Status
- Women Status
- Vaccination Created Date
- Vaccination Last Edited Date

Vaccine Information

- Vaccine ID
- Vaccine Name
- Enrollment Vaccine ID
- Vaccination Event Type
- EPI Number

Vaccination Centre Information

- Vaccination Centre ID
- Vaccination Centre Identifier

- Vaccination Centre Name
- UC
- UC ID
- Town
- Town ID
- District
- District ID
- Division
- Division ID
- Geographic Ancestry

Vaccinator Information

- Vaccinator ID
- Vaccinator Identifier
- Vaccinator Name

Immunization Information

- Immunized by LHW Flag

GPS Tracking Information

The ETL links vaccination records with vaccinator tracking data captured on the vaccination date and derives:

- Longitude
- Latitude

Output

The ETL populates:

- unfepi_dwh.dmp_women_vaccine

Each record represents a vaccination event for an enrolled woman.
Key output attributes include:

- Women ID
- Vaccination Record Number
- Vaccination Date
- Vaccination Status
- Vaccine
- Vaccination Centre
- Vaccinator
- Vaccination Event Type
- EPI Number
- Women Status
- Immunized by LHW

- Geographic Hierarchy
- GPS Coordinates
- Created Timestamp

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- Existing warehouse records for the configured refresh period are deleted before rebuilding.
- Each processed day is committed independently.
- If processing fails before a day's commit, that day's transaction is rolled back.
- Previously committed days remain intact and do not require rebuilding.

Incremental Refresh

- Warehouse records are rebuilt only for affected vaccination audit dates.
- If processing fails before the final commit, all uncommitted changes are rolled back.
- Because ETL execution status is tracked, the same affected dates are processed again during the next successful execution.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding the Women Vaccination fact table.
- Applying business rule changes.
- Recovering from historical data inconsistencies.
- Rebuilding downstream warehouse tables that depend on women vaccination data.

The processor rebuilds vaccination records from the configured historical start date through the current date.

Incremental Refresh

Used during normal scheduled ETL execution.
Only vaccination records modified since the previous successful ETL execution are recalculated, significantly reducing processing time while keeping the warehouse synchronized with operational women vaccination data.

Metadata Builder Unfepi ETL

Script Name: metadata_builder_unfepi.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi.DMP_Metadata_Builder
Process: Should run after metadata_builder_unfepi_dwh.py

Description

The Metadata ETL rebuilds supporting metadata tables and applies operational data maintenance updates required by the EPI application and downstream ETL processes.
Unlike the fact and dimension ETLs, this processor does not perform incremental processing. It executes a complete rebuild of metadata objects and applies a series of update rules each time it runs.
The processor performs the following operations:

- Rebuilds the Location Hierarchy Ancestor table.
- Rebuilds Vaccine Gap lookup tables.
- Updates missing Address location identifiers.
- Applies Leave Management void rules.
- Updates Women enrollment dates.
- Refreshes Location descriptions.
- Automatically voids expired Campaigns.

These metadata objects are used throughout reporting, warehouse ETLs, and operational application logic.

Data Sources

The ETL reads and updates the following tables.

Object	Purpose
unfepi.location	Source for rebuilding the location hierarchy and updating location descriptions.
unfepi.location_hierarchy_ancester	Rebuilt hierarchy table used throughout reporting.
unfepi.vaccinegap	Source for vaccine gap definitions.
unfepi.vaccineprerequisite	Vaccine prerequisite relationships.

unfepi.vaccine_gap_v	Rebuilt vaccine gap lookup table.
unfepi.vaccine_prereq_gap_v	Rebuilt prerequisite gap lookup table.
unfepi.address	Updated with missing Town and UC identifiers.
unfepi.leavemanagement	Leave records automatically voided according to business rules.
unfepi.publicholiday	Used to identify public holidays.
unfepi.women	Updated with enrollment dates when missing.
unfepi.womenvaccination	Source of first vaccination dates for women.
unfepi.campaign	Expired campaigns are automatically voided.

Full Refresh

The Metadata ETL always performs a complete refresh.

Process

The processor executes the following operations sequentially:

1. Rebuild location_hierarchy_ancester.
2. Rebuild vaccine_gap_v.
3. Rebuild vaccine_prereq_gap_v.
4. Update missing Address Town IDs.
5. Update missing Address UC IDs.
6. Apply Leave Management void rules.
7. Populate missing Women enrollment dates.
8. Refresh Location descriptions.
9. Void expired Campaigns.
10. Commit the transaction.

Unlike warehouse fact tables, no date-based incremental processing is performed.

Processing Logic

1. Location Hierarchy Rebuild

The ETL completely rebuilds:

- unfepi.location_hierarchy_ancester

The process:

- Deletes all existing hierarchy records.
- Calculates ancestry using GETAncestry(locationId).
- Produces every ancestor relationship between locations.
- Stores:
    - Relative Location
    - Relative Name
    - Relative Location Type
    - Parent
    - Ancestry
    - Active/Void status

This hierarchy is used extensively by downstream warehouse ETLs.

2. Vaccine Gap Lookup

The ETL rebuilds:

- unfepi.vaccine_gap_v

Gap values are converted into days.
Conversion rules:

Time Unit	Converted Gap
Day	Value
Week	Value × 7
Month	Value × 30
Year	Value × 365

Only records where:

- vaccineGapTypeId = 1

are included.

3. Vaccine Prerequisite Gap Lookup

The ETL rebuilds:

- unfepi.vaccine_prereq_gap_v

Gap values are converted into days using the same conversion rules.
Only records where:

- vaccineGapTypeId = 2

are included.

4. Address Updates

The ETL populates missing location identifiers within:
unfepi.address

Town ID

If townId is null:

- Match Town Name with location.name.
- Populate the corresponding locationId.

UC ID

If ucId is null:

- Match UC Name with location.name.
- Populate the corresponding locationId.

5. Leave Management Rules

The processor automatically voids leave records.

Public Holidays

Leave records are voided when:

- leaveDate matches a Public Holiday.

The following fields are updated:

- voided = TRUE
- voidedReason = 'Public Holiday'
- voideDate = CURDATE()

Sundays

Leave records occurring on Sundays are also automatically voided.
The following fields are updated:

- voided = TRUE
- voidedReason = 'auto voided, Sunday'
- voideDate = CURDATE()

6. Women Enrollment Update

For women whose enrollment date is missing:

- Locate the first vaccination (isFirstVaccination = TRUE).
- Update:

women.dateEnrolled = first vaccination date
This ensures enrollment dates remain available for downstream reporting.

7. Location Description Refresh

For locations of type 12 (Vaccination Centres), the ETL rebuilds the description using the location hierarchy.
The description is constructed as:
Division - District - Town - UC
This provides a standardized descriptive label for operational use.

8. Campaign Maintenance

Campaigns whose:

- endDate < CURDATE()

are automatically updated to:

- voided = TRUE
- voidedDate = CURDATE()

This prevents expired campaigns from remaining active.

Output

The ETL updates or rebuilds the following objects:

- unfepi.location_hierarchy_ancester
- unfepi.vaccine_gap_v
- unfepi.vaccine_prereq_gap_v
- unfepi.address
- unfepi.leavemanagement
- unfepi.women
- unfepi.location
- unfepi.campaign

These objects are subsequently consumed by multiple operational processes and warehouse ETLs.

Error Handling

The processor executes all operations within a database transaction.

- Metadata rebuilds are executed sequentially.
- If any operation fails before commit, the transaction is rolled back.
- No partial metadata updates are retained.

- A successful execution commits all metadata changes together, ensuring consistency across dependent tables.

Usage

The Metadata ETL is typically executed:

- During initial system deployment.
- Before warehouse ETLs that depend on refreshed metadata.
- After changes to location hierarchy.
- After updates to vaccine scheduling rules.
- After loading new public holidays.
- Following data corrections affecting operational metadata.

Because the processor rebuilds metadata and applies operational maintenance rules in a single execution, it helps ensure that both operational and warehouse processes use consistent and up-to-date reference data.

Metadata Builder Unfepi DWH ETL

Script Name: metadata_builder_unfepi_dwh.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_Metadata_Builder
Process: Should run before metadata_builder_unfepi.py

Description

The Metadata ETL populates and refreshes warehouse metadata and dimension tables required by downstream reporting and analytical ETLs.
The ETL rebuilds location hierarchy metadata, vaccine gap reference tables, vaccination centre hierarchy, HPV reporting dimensions, and vaccinator dimension tables.
The ETL supports one execution mode:
Full Refresh – Rebuilds all metadata tables completely from operational source tables.
This processor ensures warehouse reference data remains synchronized with the source system and provides consistent metadata for downstream fact and summary ETLs.

Data Sources

The ETL reads data from the following tables:

Object	Purpose
unfepi.location_hierarchy_ancester	Source location hierarchy relationships used to populate DWH hierarchy metadata.
unfepi.vaccine_gap_v	Vaccine scheduling gap reference data.
unfepi.vaccine_prereq_gap_v	Vaccine prerequisite gap reference data.
unfepi.location	Location master data including vaccination centre attributes and void information.
unfepi.vaccinator	Vaccinator demographic and assignment information.
unfepi.identifier	Vaccinator EPI identifiers.

unfepi.hpv_targets	HPV vaccination target allocation data.

unfepi_dwh.location_hierarchy_ancester	DWH hierarchy source used to generate vaccination centre hierarchy and location dimensions.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of warehouse metadata tables.

Process

Rebuild Location Hierarchy

Delete existing records from:
unfepi_dwh.location_hierarchy_ancester
Copy location hierarchy data from:
unfepi.location_hierarchy_ancester
Insert rebuilt hierarchy records into the warehouse table.

Rebuild Vaccine Gap Metadata

Delete existing records from:
unfepi_dwh.vaccine_gap_v
Reload vaccine gap definitions from:
unfepi.vaccine_gap_v
This table provides vaccine interval metadata used by downstream vaccination logic.

Rebuild Vaccine Prerequisite Gap Metadata

Delete existing records from:
unfepi_dwh.vaccine_prereq_gap_v
Reload prerequisite gap definitions from:
unfepi.vaccine_prereq_gap_v
This table provides prerequisite vaccine scheduling rules.

Rebuild Vaccination Centre Hierarchy

Delete existing records from:
unfepi_dwh.vcenter_hierarchy
Generate vaccination centre hierarchy by combining location hierarchy relationships.
The ETL derives:

- Vaccination centre

- UC
- Town
- District
- Division
- Province
- Location ancestry
- Centre identifiers
- Void status

Insert the generated hierarchy records into:
unfepi_dwh.vcenter_hierarchy

Rebuild HPV Vaccinator Dimension

Create the table if it does not exist:
unfepi_dwh.hpv_dim_vaccinator
Populate vaccinator dimension information from operational vaccinator and target tables.
The ETL derives:

- Vaccinator identifier
- EPI identifier
- Vaccinator name
- UC assignment
- Fixed vaccination target
- Community vaccination target
- School vaccination target
- Total vaccination target

Existing records are updated using duplicate key handling.

Rebuild HPV Location Dimension

Create the table if it does not exist:
unfepi_dwh.hpv_dim_location
Aggregate HPV vaccination targets at UC level.
The ETL derives:

- UC details
- District mapping
- Fixed targets
- Community targets
- School targets
- Total targets

Existing records are updated using duplicate key handling.

Rebuild Vaccinator Dimension

Delete existing records from:
unfepi_dwh.dmp_vaccinator_dim
Reload vaccinator dimension information from:

- unfepi.vaccinator
- unfepi.identifier

Insert refreshed vaccinator dimension records.

Commit Transaction

After all metadata tables are rebuilt successfully:

- Commit the transaction.

If any step fails:

- Roll back all uncommitted changes.

Aggregation Logic

The ETL performs metadata transformations for the following dimensions:

Location Hierarchy

The ETL maintains hierarchical relationships between:

- Division
- Province
- District
- Town
- Union Council
- Vaccination Centre

The hierarchy is used by reporting ETLs requiring geographic analysis.

Vaccination Centre Hierarchy

The ETL maps vaccination centres to administrative levels.
Each vaccination centre record includes:

- Vaccination centre ID
- Vaccination centre name
- UC information
- Town information
- District information
- Division information
- Province information
- Location ancestry
- Void status

HPV Vaccinator Dimension

The ETL aggregates HPV vaccination targets by vaccinator.
The ETL derives:

- Fixed session targets
- Community session targets

- School session targets
- Total vaccination targets

HPV Location Dimension

The ETL aggregates vaccinator targets at UC level.
The ETL derives:

- UC target allocation
- District mapping
- Fixed vaccination targets
- Community vaccination targets
- School vaccination targets
- Total vaccination targets

Output

The ETL populates:

- unfepi_dwh.location_hierarchy_ancester
- unfepi_dwh.vaccine_gap_v
- unfepi_dwh.vaccine_prereq_gap_v
- unfepi_dwh.vcenter_hierarchy
- unfepi_dwh.hpv_dim_vaccinator
- unfepi_dwh.hpv_dim_location
- unfepi_dwh.dmp_vaccinator_dim

unfepi_dwh.location_hierarchy_ancester

Each record represents a relationship between a location and its related parent locations.
Key output attributes include:

- Location ID
- Location name
- Relative location ID
- Relative location name
- Location type
- Relative location type
- Location ancestry
- Parent ID
- Void status

unfepi_dwh.vcenter_hierarchy

Each record represents a vaccination centre mapped to its administrative hierarchy.
Key output attributes include:

- Vaccination centre ID
- Vaccination centre name

- UC
- Town
- District
- Division
- Province
- Ancestry
- Centre identifier
- Void status

unfepi_dwh.hpv_dim_vaccinator

Each record represents a vaccinator participating in HPV activities.
Key output attributes include:

- Vaccinator ID
- EPI ID
- Vaccinator name
- UC ID
- Fixed target
- Community target
- School target
- Total target

unfepi_dwh.hpv_dim_location

Each record represents HPV target allocation at UC level.
Key output attributes include:

- UC ID
- UC name
- District ID
- District name
- Fixed target
- Community target
- School target
- Total target

unfepi_dwh.dmp_vaccinator_dim

Each record represents a vaccinator dimension entry.
Key output attributes include:

- Vaccinator ID
- Identifier
- Vaccination centre ID
- Vaccinator name
- Organisation ID
- Void status

- Void date
- System timestamp

Error Handling

The processor executes all database modifications within a transaction.

Full Refresh

All metadata tables are rebuilt during execution.
Each destination table is cleared before repopulation.
If all rebuild operations complete successfully:

- The transaction is committed.

If any operation fails:

- The transaction is rolled back.
- Previously committed metadata remains unchanged.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding warehouse metadata tables.
- Updating location hierarchy information.
- Refreshing vaccine scheduling metadata.
- Applying changes to HPV target allocations.
- Recovering from metadata inconsistencies.

The processor completely rebuilds metadata and dimension tables from operational source data.

Incremental Refresh

Not supported.
The processor performs a complete deterministic rebuild on every execution to ensure all metadata relationships and dimensions remain synchronized with source data.

Outside UC Aggregate ETL

Script Name: outside_uc_aggregate.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: DMP_OutsideUCAggregate
Process: Should run after child_vaccination_aggregate.py

Description

The Outside UC Aggregate ETL populates the unfepi_dwh.dmp_aggregrate_shifted_outside_uc summary table, which stores aggregated vaccination statistics for children who received vaccinations outside their registered Union Council (UC).
A vaccination is considered shifted outside UC when the child's registered UC differs from the UC associated with the vaccination centre where the vaccination was administered.
The ETL supports two execution modes:
Full Refresh – Rebuilds the complete shifted outside UC aggregate table from the configured historical start date.
Incremental Refresh – Rebuilds only vaccination dates affected since the last successful ETL execution.
This processor minimizes daily processing by recalculating only affected vaccination dates while allowing complete historical rebuilding when required.

Data Sources

The ETL reads data from the following tables:

Object	Purpose
unfepi_dwh.dmp_child_biography	Child demographic information including registered UC and gender.
unfepi_dwh.dmp_child_vaccine_incentive_reminder	Vaccination records, vaccination dates, vaccines, vaccination age, vaccination status, and vaccination centres.
unfepi_dwh.location_hierarchy_ancester	Vaccination centre hierarchy mapping used to identify vaccination centre UC.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of the shifted outside UC aggregate table.

Process

1. Truncate:

unfepi_dwh.dmp_aggregrate_shifted_outside_uc

2. Execute a single INSERT INTO ... SELECT statement.
3. Aggregate vaccination records from the configured historical start date.
4. Identify vaccinations where the child's registered UC differs from the vaccination centre UC.
5. Insert calculated shifted outside UC aggregates.
6. Commit the transaction.

The full refresh performs aggregation directly within MySQL, avoiding Python-side processing and reducing application overhead.

Incremental Refresh

The incremental process rebuilds only affected vaccination dates.

Step 1 – Identify affected dates

Affected vaccination dates are collected from:

- Vaccination records created after the previous successful ETL execution.
- Only vaccination records where:

vaccinationStatus = 'VACCINATED'
are considered.

Step 2 – Process affected dates

For each affected vaccination date:

1. Delete existing records from:

unfepi_dwh.dmp_aggregrate_shifted_outside_uc
for that vaccination date.

2. Recalculate shifted outside UC vaccination aggregates.
3. Insert newly generated aggregate records.
4. Commit after all affected dates are processed successfully.

This approach ensures the aggregate table remains synchronized when vaccination records are added, updated, or removed.

Aggregation Logic

The ETL identifies vaccinations where:

- The child's registered UC (child.ucId) is different from the vaccination centre UC.
- Vaccination status is one of:
    - VACCINATED
    - RETRO
    - RETRO_DATE_MISSING

The ETL aggregates vaccination activity by:

- Child registered UC
- Vaccination date
- Vaccine ID

The ETL derives shifted outside UC vaccination counts by:

Age Groups

0–11 Months

- Male shifted outside UC vaccinations
- Female shifted outside UC vaccinations

12–23 Months

- Male shifted outside UC vaccinations
- Female shifted outside UC vaccinations

24–59 Months

- Male shifted outside UC vaccinations
- Female shifted outside UC vaccinations

Output

The ETL populates:
unfepi_dwh.dmp_aggregrate_shifted_outside_uc
Each record represents shifted outside UC vaccination activity for a unique combination of:

- Child registered UC
- Vaccination date
- Vaccine

Key output attributes include:

- UC ID
- Vaccination date
- Vaccine ID
- Shifted outside UC male vaccinations (0–11 months)
- Shifted outside UC female vaccinations (0–11 months)
- Shifted outside UC male vaccinations (12–23 months)
- Shifted outside UC female vaccinations (12–23 months)
- Shifted outside UC male vaccinations (24–59 months)
- Shifted outside UC female vaccinations (24–59 months)

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

The destination table is truncated before rebuilding.
A single INSERT ... SELECT statement repopulates the aggregate table.
If the insert operation fails:

- The transaction is rolled back where applicable.

Note: In MySQL, TRUNCATE TABLE causes an implicit commit, therefore the table remains empty if the subsequent insert fails.

Incremental Refresh

Aggregate records are rebuilt only for affected vaccination dates.
For each affected date:

- Existing records are deleted.
- Aggregates are recalculated.
- New records are inserted.

If processing fails:

- The transaction is rolled back.
- Previously committed data remains unchanged.

Because the ETL records execution status, the same affected vaccination dates are processed again during the next successful run.

Deployment Fallback

If the initial full-refresh deployment fails after TRUNCATE TABLE has executed:

- Restore the contents of dmp_aggregrate_shifted_outside_uc using the existing recovery process before retrying the ETL deployment.

Since TRUNCATE performs an implicit commit in MySQL, the table cannot be restored through transaction rollback.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding historical shifted outside UC aggregates.
- Applying business rule changes.
- Recovering from historical data inconsistencies.

The processor rebuilds shifted outside UC vaccination aggregates from the configured historical start date.

Incremental Refresh

Used during normal scheduled ETL execution.
Only vaccination dates affected since the previous successful ETL run are recalculated, reducing processing time while keeping shifted outside UC statistics synchronized with vaccination activity.

Women Defaulter ETL

Script Name: women_defaulter.py
Current Deployment: Staging (dev branch)
Full Refresh Stored Procedure: unfepi_dwh.DMP_WomenDefaulter

Description

The Women Defaulter ETL populates the unfepi_dwh.dmp_women_defaulter fact table, which identifies women who have defaulted on scheduled vaccination doses.
A woman is considered a defaulter when a scheduled vaccine dose remains incomplete beyond the defined default period. The ETL evaluates vaccine prerequisites and vaccination schedules to identify missed or delayed doses for women enrolled in the vaccination program.
The ETL supports two execution modes:
Full Refresh – Rebuilds the complete defaulter fact table from historical warehouse data.
Incremental Refresh – Rebuilds records only for women affected since the last successful ETL execution.
This processor minimizes daily processing by recalculating only impacted women while allowing a complete rebuild whenever business rules change or historical vaccination data requires correction.

Data Sources

The ETL reads data from the following tables/views:

Object	Purpose
unfepi_dwh.dmp_women_biography	Women demographic information, enrollment details, birth dates, and enrollment vaccine information.
unfepi_dwh.dmp_women_vaccine	Women vaccination records, vaccination status, due dates, vaccination dates, centers, and vaccinators.
unfepi_dwh.vaccine_prereq_gap_v	Vaccine prerequisite mapping and vaccine schedule gap definitions.
unfepi.womendailyaudit	Identifies women whose vaccination information has changed since the previous ETL execution.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of the women defaulter fact table.

Process

1. Truncate unfepi_dwh.dmp_women_defaulter.
2. Execute the defaulter calculation query.
3. Process generated records in configurable batches.
4. Insert calculated defaulter records into the destination table.
5. Commit the transaction.

The full refresh performs the complete defaulter calculation from historical warehouse data while using batch inserts to reduce memory consumption during large rebuilds.

Incremental Refresh

The incremental process rebuilds only affected women.

Step 1 – Identify affected women

Affected women IDs are collected from three sources:

- Women vaccination records created after the previous successful ETL execution.
- Entries in unfepi.womendailyaudit modified after the previous successful execution.
- Women whose vaccination due dates have entered the 28-day default window since the last execution.

The processor returns affected womenId values and recalculates only those women.

Step 2 – Process affected women

For each batch of affected women IDs:

1. Delete existing records for those women from unfepi_dwh.dmp_women_defaulter.
2. Recalculate defaulter information.
3. Insert newly generated defaulter records.
4. Commit after all batches complete successfully.

This approach ensures the defaulter fact table remains synchronized when vaccination records are inserted, updated, or corrected.

Defaulter Identification Logic

For each enrolled woman, the ETL evaluates eligible vaccine doses based on vaccine prerequisite relationships.

A woman is considered a defaulter when:

- The vaccine dose is part of the supported maternal vaccination schedule.
- The vaccine dose is required after the enrollment vaccine.

- The vaccine due date has exceeded the allowed completion period.
- The vaccination has:
    - remained in SCHEDULED status beyond the default period, or
    - been completed after the allowed due date window.

The ETL currently evaluates TT vaccination doses:

Vaccine ID	Vaccine Name
17	TT1
18	TT2
19	TT3
20	TT4
21	TT5

The ETL derives:

- Enrollment age
- Default age
- Enrollment vaccine
- Defaulted vaccine
- Vaccination due date
- Default date
- Vaccination status
- Vaccination date
- Enrollment center
- Defaulting center
- Enrollment vaccinator
- Defaulting vaccinator

Output

The ETL populates:
unfepi_dwh.dmp_women_defaulter
Each record represents a woman–vaccine combination where the woman has defaulted according to the configured business rules.

Key output attributes include:

- Woman identifier
- Enrollment age in days
- Default age in days
- Enrollment vaccine ID
- Defaulted vaccine ID
- Defaulted vaccine name
- Enrollment date
- Default vaccination status
- Default vaccine due date
- Default date
- Vaccination date
- Enrollment center ID
- Default center ID
- Enrollment vaccinator ID
- Default vaccinator ID

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- The destination table is truncated before rebuilding.
- Defaulter records are inserted in batches.
- If processing fails, the transaction is rolled back where applicable.
- The next execution can restart the rebuild from the beginning.

Note: Since TRUNCATE TABLE causes an implicit commit in MySQL, the destination table remains empty if the subsequent insert operation fails.

Incremental Refresh

- Women records are processed in configurable batches.
- Existing records for affected women are deleted before recalculation.
- If any batch fails, all uncommitted changes are rolled back.
- Because ETL execution status is recorded, the same affected women IDs are processed again during the next successful run.

Deployment Fallback

If the initial full-refresh deployment fails after TRUNCATE TABLE has executed, restore the contents of:
unfepi_dwh.dmp_women_defaulter
using the existing rebuild process or stored procedure (if available) before retrying the ETL deployment.

Since TRUNCATE TABLE automatically commits in MySQL, the table cannot be restored through transaction rollback.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding the women defaulter table after business rule changes.
- Recovering from historical vaccination data inconsistencies.

The processor truncates the destination table and reconstructs all women defaulter records from warehouse source tables.

Incremental Refresh

Used during normal scheduled ETL execution.
Only women affected since the previous successful ETL run are recalculated, reducing processing time while keeping the defaulter fact table synchronized with operational vaccination data.

Women Performance Indicators ETL

Script Name: women_performance_indicators.py
Current Deployment: Production (main branch)
Full Refresh Stored Procedure: N/A
Process: New ETL for populating and incrementally updating summary tables

Description

The Women Vaccination Summary ETL populates the unfepi_dwh.dmp_summary_women_vac summary table, which provides aggregated vaccination statistics for women vaccination activities.
The ETL summarizes completed vaccination records by vaccination date, vaccination center, vaccinator, vaccine, and vaccination event type. The output is used for operational reporting and dashboard analytics.
The ETL supports two execution modes:
Full Refresh – Rebuilds vaccination summary records from the configured historical start date through the current date.
Incremental Refresh – Rebuilds summary records only for vaccination dates affected since the last successful ETL execution.
This processor minimizes daily processing by recalculating only affected vaccination dates while allowing complete historical rebuilding when required.

Data Sources

The ETL reads data from the following tables:

Object	Purpose
unfepi_dwh.dmp_women_vaccine	Source vaccination records including vaccination date, vaccine, vaccination status, center, vaccinator, and event type.
unfepi_dwh.dmp_women_biography	Women enrollment information used to calculate enrollment counts.

Full Refresh

The full refresh is intended for initial deployments or complete rebuilding of the women vaccination summary table.

Process

1. Determine the refresh period from full_refresh_start_date through the current date.
2. Delete existing records from unfepi_dwh.dmp_summary_women_vac where vaccination date is within the refresh range.
3. Process each vaccination date individually.
4. Aggregate vaccination summary information for the selected date.
5. Insert calculated summary records.
6. Commit each successfully processed date.

The processor rebuilds summaries day by day to reduce memory consumption and support large historical refresh periods.

Incremental Refresh

The incremental process rebuilds only affected vaccination dates.

Step 1 – Identify affected dates

Affected vaccination dates are collected from:

- Vaccination records created after the previous successful ETL execution.
- Only records where:

vaccinationstatus = 'VACCINATED'

- Only dates greater than or equal to the configured full refresh start date are considered.

Step 2 – Process affected dates

For each affected vaccination date:

1. Delete existing records for that date from dmp_summary_women_vac.
2. Recalculate vaccination summary information.
3. Insert newly generated summary records.
4. Commit after all affected dates are processed successfully.

This approach ensures summary data remains synchronized whenever vaccination records are added or updated.

Aggregation Logic

The ETL aggregates completed women vaccination activities by:

- Vaccination date
- Vaccination center
- Vaccinator
- Vaccine
- Vaccination event type

Event Type Normalization

Vaccination events are normalized into two reporting categories:

OUTREACH

Includes:

- EVENING-OUTREACH
- MORNING-OUTREACH
- MORNING-VAN
- EVENING-VAN
- OUTREACH
- VAN
- CAMPAIGN

FIXED

Includes:

- MORNING-FIXED
- EVENING-FIXED
- FIXED
- -FIXED

The ETL derives:

- Number of enrollments
- Number of immunizations

Enrollment counts are calculated by matching:

- Woman enrollment records
- Enrollment vaccine
- Enrollment date
- Vaccination date

Only completed vaccination records are included:
vaccinationStatus = 'VACCINATED'

Vaccination records without a vaccinator are excluded.

Output

The ETL populates:
unfepi_dwh.dmp_summary_women_vac

Each record represents aggregated vaccination activity for a unique combination of:

- Vaccination center
- Vaccinator
- Vaccination event type
- Vaccination date

- Vaccine

Key output attributes include:

- Center ID
- Vaccinator ID
- Vaccination event type
- Vaccination date
- Vaccine ID
- Enrollment count
- Immunization count

Error Handling

The processor executes all database modifications within transactions.

Full Refresh

- Existing summary records for the refresh period are deleted before rebuilding.
- Vaccination summaries are rebuilt date by date.
- Each successfully processed date is committed independently.
- If processing fails, uncommitted changes are rolled back.
- Previously committed dates remain intact.

Incremental Refresh

- Summary records are rebuilt only for affected vaccination dates.
- Existing records for affected dates are deleted before recalculation.
- If processing fails before commit, changes are rolled back.
- Because ETL execution status is recorded, the same affected dates are processed again during the next successful run.

Deployment Fallback

If the initial full-refresh deployment fails after the delete operation has started, restore the contents of:
unfepi_dwh.dmp_summary_women_vac

before retrying the ETL deployment.
The table contents cannot be recovered through rollback if committed deletion operations have already completed.

Usage

Full Refresh

Used when:

- Deploying the ETL for the first time.
- Rebuilding historical vaccination summaries.
- Applying changes to aggregation rules.
- Recovering from historical vaccination data inconsistencies.

The processor rebuilds all women vaccination summaries from the configured historical start date through the current date.

Incremental Refresh

Used during normal scheduled ETL execution.
Only vaccination dates affected since the previous successful ETL run are recalculated, reducing processing time while keeping the summary table synchronized with operational vaccination data.

Order of Execution:

The scripts should be scheduled to run in the following order:

ETL Script Name	Replaced Stored Procedure
1. metadata_builder_unfepi_dwh 2. metadata_builder_unfepi	unfepi_dwh.DMP_Metadata_Builder();
3. master_child_data 4. master_child_data_bio	unfepi_dwh.DMP_MasterChildData();
5. master_women_data 6. master_women_data_bio	unfepi_dwh.DMP_MasterWomenData();
7. child_defaulter	unfepi_dwh.DMP_ChildDefaulter();
8. women_defaulter	unfepi_dwh.DMP_WomenDefaulter();
9. child_vaccination_aggregate	unfepi_dwh.DMP_ChildVaccinationAggregate();
10. outside_uc_aggregate	unfepi_dwh.Outside_UC_Aggregate
11. child_vaccination_aggregate_monthly	unfepi_dwh.DMP_ChildVaccinationAggregateMonthly();
12. child_performance_indicators.py	N/A
13. women_performance_indicators.py	N/A
14. hpv.py	N/A
15. In Progress	unfepi_dwh.ETL_Attendance();

16. To Do	unfepi_dwh.ETL_Compliance_New();

Remaining Stored Procedures to be converted:

ETL Epic: https://seirpk.atlassian.net/browse/ZMMDR-841
Remaining Stored Procedures:

- unfepi_dwh.DMP_PerRowChildData();
- unfepi_dwh.DMP_PerRowWomanData();
- unfepi_dwh.ETL_Compliance_New();
- unfepi_dwh.ETL_Attendance(); (In Progress)