DMS Tool

User Guide

Comprehensive Guide to Using the Data Management System

Version 4.0.0

DMS Tool – End■to■End User Guide

Table of Contents

---

1. Introduction

1.1 What the DMS Tool Is

The **DMS Tool** (Data Management System / Data Warehouse Tool) is a web application designed to:

- **Extract** data from one or more **source systems** (Oracle, PostgreSQL, and others via connections).
- **Transform** that data using **reusable mapping rules**.
- **Load** the results into **target data warehouse tables** (dimensions, facts, staging).
- **Schedule and monitor** recurring jobs that execute those mappings.
- **Provide visibility** into job runs, errors, and overall health through dashboards and logs.

You can think of the DMS Tool as a **visual, governed ETL/ELT front end** on top of your enterprise data warehouse.

1.2 Who This Guide Is For

This guide is written for:

- **Business / data analysts** who design mappings and jobs but may not write low■level code.
- **Data engineers** who need to understand how to use the UI correctly and how data flows.
- **Admins / power users** who configure users, roles, and connections.

You do **not** need to know Python or PL/SQL to use the application, but you should:

- Understand your **source system tables** and **target data model**.
- Be comfortable reading/writing **SQL** where needed (especially in Manage SQL).

1.3 High■Level Data Flow

At a high level, every use of the tool follows the same pattern:

```
Source DB / Files ■ ■ (1) Read via a Connection + SQL (Manage SQL, File Upload) ▼
Mapping Rules (Data Mapper) ■ ■ (2) Apply transformations, mapping, SCD rules ▼
Target Tables in DW (CDR schema) ■ ■ (3) Scheduled / manual Jobs run mappings
repeatedly ▼ Dashboards, Reports, and Downstream Consumers
```

The rest of this guide explains **each part of this flow**, and how you, as a user, interact with it through the UI.

---

2. Getting Started

2.1 Accessing the Application

- Open a modern browser (Chrome, Edge, Firefox).
- Navigate to your environment URL, for example:

- `http://localhost:3000` (local/development)

- `https://dms.your-company.com` (production)

- You will be redirected to the **Login** page under `/auth/login`.

> If you do not have a user account, contact your **DMS administrator**.

2.2 First Login & Password Change

1. **Login**

- Enter your **username** and **password**.

- If reCAPTCHA is enabled, tick “I’m not a robot”.

- Click **Sign In**.

2. **Forced Password Change (if configured)**

- For brand■new or reset accounts, you may be forced to change your password.

- You’ll be redirected to `/auth/change-password`.

- Enter:

- Current password.

- New password (follows your org's complexity rules).

- Confirm new password.

- Click **Update Password**.

3. After successful login you will be redirected to the **Home** page.

If login fails, you'll see a friendly error like:

- "Hmm, that didn't work. Try again!" for invalid credentials.
- "Account locked. Please try again after 15 minutes" if there were too many failed attempts.

2.3 Understanding the Home Screen & Navigation

After login you typically land on `/home`.

Common elements you'll see:

- **Top Bar**

- Product name/logo.

- Current environment (for example: DEV / UAT / PROD).

- User profile menu (top right) – access **Profile**, **Change Password**, and **Logout**.

- **Left Sidebar**

- A list of **modules** you can access, for example:

- Home

- Register DB Connections

- Manage SQL

- Type Mapper

- Mapper Module

- Jobs

- Job Status & Logs

- Dashboard

- Reports, Report Runs

- File Upload

- Security (for admins)

- Admin (for admins)

- Modules shown here are controlled by:

- Your **role** (admin, power user, normal user).

- The **Security** module configuration (which modules are enabled per user).

- **Main Content Area**

- Contextual content based on selected module.

- Summary cards, tables, charts, forms, etc.

> If you see a “Not Authorized” message when clicking a module, your user or role does not have access to that module. Contact your administrator.

---

3. Core Concepts & Terminology

Before diving into the screens, it helps to understand a few core ideas.

3.1 Source Systems, Targets, and Connections

- **Source System**

Any system from which data is read:

- ERP database (Oracle, PostgreSQL, etc.).

- Source data mart.

- Flat files (via File Upload).

- **Target**

Where data ends up in the data warehouse:

- **CDR schema** tables such as `DIM_CUSTOMER`, `FACT_SALES`, staging tables.

- **Connection**

A reusable set of credentials and network details that lets the DMS Tool talk to a database:

- Defined once in **Register DB Connections**.

- Reused in:

- **Manage SQL**: to run source queries.

- **Mapper Module**: to define the **target connection** for loads.

- **Jobs** and **File Upload**: to read/write data.

Think of **connections** as “saved logins” to databases that the rest of the tool can safely reference.

3.2 Mappings and Jobs

- **Mapping**

A mapping defines **how data should look** in the target:

- Which source columns feed which target columns.

- What transformations are applied (e.g., concatenation, lookups, CASE expressions).

- Whether history is tracked (SCD Type 2) or overwritten (SCD Type 1).

- Which table type: **Dimension**, **Fact**, or **Staging**.

- **Job**

A job defines **when and how a mapping (or report) is executed**:

- Which mapping to run.

- How often (daily, weekly, monthly, interval).

- Any parameters (date range, history load, etc.).

- Whether there are dependencies on other jobs.

3.3 Schedules, Runs, and Logs

- **Schedule**

A configuration that says, “Run job X at frequency Y”, for example:

- Daily at 2 AM.

- Every 15 minutes.

- Every Monday at 9 AM.

- **Run / Execution**

A single execution of a job:

- Has start time, end time, status, and row counts.

- Is visible in **Job Status & Logs** and the **Dashboard**.

- **Logs**

Detail of what happened during a run:

- Progress messages.

- Error messages.

- Row counts and performance information.

3.4 Roles, Permissions, and Licenses

- **Roles**: Group permissions, such as Admin, Power User, Viewer.
- **Permissions**: Control which modules and actions a user can perform.
- **License**: Controls whether the application is activated and for how long.

Admins manage all of this under the **Admin** and **Security** modules.

---

4. Typical End■to■End Workflow (Big Picture)

This section gives you a “map in your head” of how the application is meant to be used.

4.1 From Source Data to Dashboard – Step■By■Step

1. **Register DB Connections**

- Define connections to:

- The metadata database (DMS schema).

- Source databases.

- Target databases (CDR schema).

2. **Author Source SQL**

- In **Manage SQL**, create reusable **SQL codes** that extract the data you want from the source.

- Validate and test queries.

3. **Define Mappings**

- In **Mapper Module**, create a **mapping**:

- Choose table type (Dimension, Fact, Staging).

- Configure target table and schema.

- Define each target column and the logic that populates it.

- Link to one or more SQL codes from Manage SQL as needed.

4. **Create Jobs**

- In **Jobs**, create a job that runs your mapping:

- Configure whether it is a one■off, daily, weekly, etc.

- Optionally define dependencies (Job B waits for Job A).

5. **Enable Scheduler**

- Ensure the **scheduler service** is running (usually handled by IT).

- Once running, it will:

- Synchronize job schedules.

- Pick up immediate run requests.

- Execute job flows using the execution engine.

6. **Monitor Job Status**

- Use **Job Status & Logs** to:

- See each run.

- View logs and error details.

- Stop a run if needed.

7. **Review Dashboards and Reports**

- Use the **Dashboard** to see overall system health and performance.

- Use **Reports** to define and **Report Runs** to access generated reports.

4.2 Example: Daily Customer Dimension Load

Imagine you want a **Customer Dimension** table that is refreshed daily at 2 AM:

1. Define an **Oracle connection** to your ERP system (source).
2. Define a **PostgreSQL connection** to your DW (target).
3. In **Manage SQL**, create `SQL_CUSTOMER_SRC` that selects customer data from ERP.

4. In **Mapper Module**, create mapping `MAP_CUSTOMER_DIM`:

- Table type: Dimension (SCD Type 2 for history).

- Target: `DIM_CUSTOMER` in `CDR` schema.

- Define mappings for `CUSTOMER_ID`, `NAME`, `ADDRESS`, `STATUS`, etc.

5. In **Jobs**, create job `JOB_CUSTOMER_DAILY`:

- Attached to mapping `MAP_CUSTOMER_DIM`.

- Schedule: Daily at 2:00 AM.

6. Ensure scheduler is running.
7. Next morning, check **Job Status & Logs**:

- Confirm run succeeded.

- See row counts and any warnings.

8. Check BI tool or the **Dashboard** to confirm updated metrics.

---

5. Module■By■Module Guide

This section explains **what each module is for** and **how to use it step■by■step**.

5.1 Authentication & Login (Auth Module)

What it is designed for

Provides secure login, logout, password change, and password reset workflows.

How it works (high level)

- Frontend collects your username, password, and optional reCAPTCHA.
- Backend validates the credentials against the **SQLite users** table.
- On success, backend issues a **JWT token** stored in localStorage and cookies.
- Subsequent requests include this token to prove your identity.

How to use it

- **Login**

1. Go to `/auth/login`.

2. Enter username and password.

3. Complete reCAPTCHA (if visible).

4. Click **Sign In**.

- **Forgot Password**

1. Click **Forgot Password** on the login page.

2. Enter your email.

3. Check email for reset link.

4. Click link and set a new password.

- **Change Password (after login)**

1. From the top■right profile menu, choose **Change Password** or go to `/auth/change-password`.

2. Enter current and new passwords.

3. Click **Update Password**.

- **Logout**

1. Click the profile menu.

2. Click **Logout**.

5.2 Home & Navigation

What it is designed for

The **Home** page and navigation help you quickly see what's important and move between modules.

How it works

- The sidebar is driven by your **permissions and enabled modules**.
- The home page may show:

- Quick links to main modules.

- Summary of recent jobs, issues, or notifications.

How to use it

1. After login, land on `/home`.
2. Use the sidebar to switch modules.
3. Use cards/links on the home page for shortcuts (for example, “Create new mapping”).

5.3 Register DB Connections

What it is designed for

To define **reusable database connections** that can be referenced by Manage SQL, Mapper, Jobs, and File Upload modules.

How it works

- Connection details (host, port, service/db name, username, password) are stored in metadata.
- Each connection has:

- Status (active/inactive).

- Type (Oracle/PostgreSQL/etc.).

- A friendly name (for example, “ERP_PROD_ORA”).

- Modules use the connection `CONID` to connect to the appropriate database.

How to use it

1. Click **Register DB Connections** in the sidebar.
2. To **add a new connection**:

- Click **Add Connection**.

- Fill in:

- Connection Name (descriptive).

- Database Type (Oracle/PostgreSQL).

- Host, Port.

- Service Name / Database Name.

- Username, Password (or connection string).

- Click **Test Connection**.

- If test passes, click **Save**.

3. To **edit a connection**:

- Click the edit icon next to a connection.

- Update details, re■test, and **Save**.

4. To **deactivate** a connection:

- Use the toggle or Deactivate button.

- Deactivated connections won’t appear for selection in other modules.

Best practice

• Use descriptive names like:

- `ERP_OLTP_ORACLE_DEV`

- `DWH_POSTGRES_PROD`

• Never use personal accounts; always use **service accounts** managed by DBAs.

5.4 Manage SQL (Source Query Management)

What it is designed for

To manage **reusable source SQL queries** that mappings can consume.

How it works

• Each SQL definition has:

- **Code** (for example, `SQL_CUSTOMER_SRC`).

- **Description**.

- **Source Connection** (from Register DB Connections).

- **SQL text**.

• These definitions are stored in metadata tables and can be used in:

- Mapper Module (as data sources).

- Reports.

How to use it

1. Go to **Manage SQL**.
2. To **create a new SQL definition**:

- Click **New SQL**.

- Fill:

- SQL Code (unique identifier).

- Description.

- Select the **Source Connection**.

- Enter SQL text in the editor.

- Click **Validate** to check syntax.

- Click **Test** to preview data (first N rows).

- Click **Save** to store it.

3. To **edit a SQL definition**:

- Click on the row or edit icon.

- Update SQL or description.

- Re■validate and **Save**.

4. To **use in a mapping**:

- In Mapper Module, choose this SQL code as your source.

Example

```
SELECT c.customer_id, c.first_name || ' ' || c.last_name AS full_name, c.status,
c.created_date FROM erp_customers c WHERE c.status = 'ACTIVE';
```

5.5 Type Mapper (Parameter & Data Type Mapping)

What it is designed for

To manage **data type mappings and parameter types** between different databases and internal representations.

How it works

- Maintains lists such as:

- Source data types vs target data types.

- Parameter data types used in mappings and jobs.

- Ensures consistent handling of types when generating code and executing jobs.

How to use it (typical user)

Most business users will rarely need to modify this. It’s primarily for:

- Data engineers.
- Administrators performing cross■database migrations.

Use cases:

- When moving from Oracle to PostgreSQL, ensure types like `NUMBER`, `VARCHAR2`, `DATE` are correctly mapped to `NUMERIC`, `VARCHAR`, `TIMESTAMP`, etc.

5.6 Data Mapper (Mapper Module)

What it is designed for

This is the **heart of the system**. It defines:

- How data flows from the source SQL to the target table.
- All column■level transformation logic.
- Whether you track history (SCD) or overwrite.

How it works (simplified)

- You define:

- **Mapping Header** (reference, description, table type, target schema, etc.).

- **Field Mappings** for each target column.

- When you save and validate, the backend:

- Stores this configuration in metadata tables.

- Can generate executable Python job flow code that the scheduler executes later.

How to use it (step■by■step)

1. Go to **Mapper Module**.
2. **View existing mappings**:

- See list with Reference, Description, Table Name, Status.

3. **Create a new mapping**:

1. Click **New Mapping**.

2. Fill header:

- **Reference**: Unique name, e.g., `MAP_CUSTOMER_DIM`.

- **Description**: “Customer dimension from ERP”.

- **Source System**: Logical name of source (ERP, CRM, etc.).

- **Table Name**: Target table, e.g., `DIM_CUSTOMER`.

- **Table Type**:

- **Dimension** – Surrogate keys, SCD.

- **Fact** – Measures and foreign keys.

- **Staging** – Raw staging area.

- **Target Schema**: e.g., `CDR`.

- **Frequency Code**: How often this mapping will typically run.

- **Bulk Process Rows**: Batch size for processing.

- **Target Connection**: Choose which DB connection is used to create/populate the table.

3. **Link source SQL**:

- Either:

- Choose a SQL code from **Manage SQL**.

- Or paste/write SQL directly (depending on configuration).

- Load columns to show available fields from the source.

4. **Configure field mappings**:

- For each target column:

- **Field Name**: Target column name.

- **Data Type**: Target data type.

- **Primary Key** and **PK Sequence** (for key columns).

- **SCD Type**:

- Type 1: Overwrite.

- Type 2: Track history (start/end dates, status).

- **Logic**: Expression that populates the field.

- **Key Column** / **Value Column** (for lookups).

- **Combine Code**: How to aggregate multiple rows (SUM, MAX, etc.).

- **Execution Sequence**: Order (helpful if outputs of one expression feed another).

5. **Validate mapping**:

- Click **Validate**:

- Checks for missing required fields.

- Checks data type consistency.

- Validates SQL and mapping expressions.

- Fix any errors highlighted.

6. **Save mapping**:

- Click **Save** or **Save to Database**.

- The system persists header and detail rows.

4. **Edit an existing mapping**:

- Open mapping, change header or fields, re■validate, and **Save**.

5. **Activate / Deactivate**:

- Activate when ready for job creation.

- Deactivate to prevent further use (existing jobs may still reference old versions depending on configuration).

How data flows through a mapping

Simplified:

```
Source Data (via SQL) ■■■ Apply each field's logic ■■■ Target Row
```

For SCD Type 2 dimensions, the engine:

- Compares hash of incoming row vs existing row.
- If different:

- Closes existing row (set end date).

- Inserts a new row (new version).

5.7 Jobs (Job Design & Scheduling)

What it is designed for

To turn mappings into **runnable jobs** that can be executed on a schedule or on demand.

How it works

• Each job references:

- A **mapping** (or report).

- Optional **time parameters** (for history runs).

- Optional **dependencies** (run after other jobs).

• Schedules are stored in metadata and synchronized to the scheduler service.

How to use it

1. Go to **Jobs**.
2. **View existing jobs**:

- See Job ID, Mapping reference, Description, Status.

3. **Create a job**:

1. Click **New Job** or **Create Job**.

2. Select a **Mapping** (only active ones).

3. Enter:

- Job description.

- Any default parameters (such as date range).

4. Configure schedule:

- Choose frequency: daily, weekly, monthly, etc.

- Set time (hour, minute) and starting date.

- Optionally end date.

5. Save job and enable schedule if you want it to auto■run.

4. **Run a job immediately**:

- From the job list, click **Run Now**.

- Confirm the execution.

5. **Edit a job**:

- Change description, schedule, or parameters.

- Save updates.

6. **Set dependencies**:

- In advanced dialogs (such as Dependency Modal), define:

- Job B depends on Job A:

- Job B won't run until Job A completes successfully.

5.8 Job Status & Logs

What it is designed for

To give you clear visibility into **what has run**, **is running**, and **why something failed**.

How it works

• Reads execution records from tables such as:

- `DMS_PRCLOG`, `DMS_JOBLOG`, `DMS_JOBERR`.

• Shows:

- Start/end time.

- Status (In Progress, Completed, Failed, Stopped).

- Row counts.

- Links to:

- Detailed logs.

- Error messages.

How to use it

1. Go to **Job Status & Logs**.
2. Use filters:

- Date range.

- Status (success/failure).

- Mapping reference.

- Job ID.

3. Click on a specific run:

- See detailed summary.

- Open logs:

- A chronological list of messages.

- Open error details:

- Error text.

- Possibly affected table/column.

4. To **stop a running job**:

- Find a run with status “In Progress”.

- Click **Stop Job**.

- Confirm.

Use this module whenever:

- A job is **red** on the dashboard.
- A downstream consumer says data is stale.
- You want to confirm if yesterday's loads were successful.

5.9 Dashboard (Monitoring & Analytics)

What it is designed for

To provide a **visual summary** of system health and performance:

- How many jobs succeed/fail.
- How long jobs take.
- How much data is processed.

How it works

- Aggregates data from job logs.
- Provides charts and KPI cards for:

- Success/failure counts.

- Average run time.

- Processed rows.

How to use it

1. Go to **Dashboard**.
2. Choose your **date range**.
3. Review:

- Success/failure charts.

- Distribution of run durations.

- Top slow jobs.

4. Use this to:

- Identify chronic failing jobs.

- Spot performance regressions.

- Confirm that overnight runs processed expected volumes.

5.10 Reports & Report Runs

What they are designed for

To define and execute **report definitions** and to access their generated outputs.

How it works

• A **Report** definition:

- Specifies data sources (SQL, mappings).

- Defines layout (columns, groupings, formatting).

- Specifies output format (PDF, Excel, CSV).

• **Report Runs**:

- Capture individual generations of a report.

- Allow users to download outputs.

How to use them

1. **Define a report** (Reports module):

- Create a new report definition.

- Choose data sources (Manage SQL or mappings).

- Configure layout and parameters.

- Save.

2. **Run a report**:

- From the report definition, click **Generate**.

- Choose parameters (date range, filters).

- Wait for completion.

3. **View past runs** (Report Runs module):

- Download existing outputs.

- Check status and timings.

5.11 File Upload Module

What it is designed for

To allow users to upload data from **files** (CSV, Excel, JSON, etc.) into the data warehouse, with repeatable column mappings and transformations.

How it works

- Upload file → Parse & inspect → Configure column mappings → Process into target.
- Supports:

- Column mapping.

- Data type enforcement.

- Basic transformations.

- Preview and validation.

How to use it

1. Go to **File Upload**.
2. **Upload a file**:

- Click **Upload File**.

- Select file from your computer.

- Provide:

- File reference.

- Description.

- Source system.

- Target connection (if data is loaded to DB).

- Click **Upload**.

3. **Configure column mappings**:

- For each detected column:

- Map to target column name.

- Choose data type.

- Mark required or optional.

- Define default values or transformations.

- Save mappings.

4. **Preview**:

- Preview a few rows to confirm data looks right.

5. **Process**:

- Choose processing options (Insert/Update/Upsert).

- Start processing.

- Monitor status and errors.

Use the File Upload module when:

- You receive data as **files** from external parties.
- You need a one■time or recurring **file■based load**.

5.12 Security (Module Access)

What it is designed for

To control **which user sees which modules** in the UI.

How it works

- Maintains a matrix of:

- Users/roles → Modules → Permissions.

- Uses a central list of modules (Manage SQL, Mapper, Jobs, etc.).
- The frontend checks this configuration to:

- Show/hide modules in the sidebar.

- Block unauthorized access.

How to use it (Admin only)

1. Go to **Security**.
2. For each user or role:

- Enable or disable modules.

3. Save changes.

This ensures that:

- Business users only see relevant modules.
- Admin and technical modules are restricted.

5.13 Admin (Users, Roles, License)

What it is designed for

Central place for admins to manage:

- Users.
- Roles and permissions.
- Licenses.
- System notifications.

How to use it

- **Users**

- Add / edit users.

- Activate / deactivate accounts.

- Reset passwords.

- **Roles**

- Define roles such as `DMS_ADMIN`, `DMS_DEVELOPER`, `DMS_USER`.

- Assign modules and permissions to roles.

- **License Manager**

- View license status and expiry.

- Enter or update license keys.

Regular users will rarely need to access this module.

5.14 Profile (Personal Settings)

What it is designed for

To let you see and update **your own profile information**.

How to use it

1. Click on your profile icon (top right).
2. Choose **Profile**.
3. See details such as:

- Name, email, department.

- Role and last login time.

4. Update allowed fields (for example, phone number).
5. Click **Save**.

Use this module to keep your contact details current.

---

6. Data Flow Details

6.1 How Data Moves Through the System

At runtime, a typical job does the following:

```
1. Scheduler decides it’s time to run Job J 2. Execution engine looks up the mapping(s) for Job J 3. For each mapping: a. Connect to source DB using connection defined in Manage SQL b. Execute the source SQL and fetch rows c. For each row, apply field mapping logic d. Connect to target DB using mapping's target connection e. Insert/update rows in the target table f. Write logs and row counts 4. Dashboard and Job Status modules read these logs for display
```

6.2 Where Data Is Stored

- **Source data**: In the **source system database** or **files**.
- **Metadata** (mappings, jobs, connections, parameters): In DMS schema tables (Oracle/PostgreSQL).
- **Target data**: In CDR schema tables.
- **User and security data**: In SQLite (for app users, roles, etc.).
- **Logs and execution history**: In metadata tables (`DMS_PRCLOG`, `DMS_JOBLOG`, `DMS_JOBERR`) and backend log files.

6.3 How Changes and History Are Tracked

- **Metadata changes**:

- Mappings and jobs are versioned logically via flags and timestamps.

- **Data changes** (SCD):

- Type 1 – overwrite.

- Type 2 – end■date old row, insert new row with a new hash and effective dates.

- **Hash column (`RWHKEY`)**:

- Stores an MD5 hash of the important data columns.

- Makes comparisons efficient (one hash vs many columns).

---

7. How To… Scenarios (Recipes)

7.1 Onboard a New Source System

1. **Register connections** to the new source DB in **Register DB Connections**.
2. Identify target tables and schemas with your data architect.
3. For each subject area (Customer, Product, etc.):

- Create a **source SQL** in **Manage SQL**.

- Create a **mapping** in **Mapper Module**.

- Create a **job** in **Jobs**.

- Test runs via **Run Now**.

- Monitor results via **Job Status & Logs**.

7.2 Build a New Mapping and Job

1. Confirm that:

- Required source tables/columns exist.

- Necessary DB connections are defined.

2. In **Manage SQL**:

- Create a new SQL definition (test it).

3. In **Mapper Module**:

- Create mapping reference.

- Define fields and transformation logic.

- Validate and save.

4. In **Jobs**:

- Create a job, link it to the mapping.

- Optionally define schedule.

5. Test via **Run Now**:

- Validate row counts in target.

- Ensure business users confirm results.

7.3 Investigate a Failed Job

1. Go to **Job Status & Logs**.
2. Filter for:

- Status = Failed.

- Target date/time window.

3. Open the **failed run**:

- Check error message summary.

- Open detailed logs for more context.

4. Typical issues:

- DB connection problems.

- Missing tables/columns.

- Data type conversion errors.

- Logic errors in mapping expressions.

5. Fix the root cause:

- In Register DB Connections.

- In Manage SQL.

- In Mapper Module.

6. Re■run the job (either **Run Now** or wait for schedule).

7.4 Safely Change Existing Logic

1. Identify impacted mapping(s) and job(s).
2. Communicate change with stakeholders (owners of reports/dashboards).
3. In **Mapper Module**:

- Open the mapping.

- Update specific field logic.

- Validate.

4. In **Jobs**:

- If needed, update parameters or schedule.

5. Test in a **non■production environment**:

- Run job on a subset or test date range.

- Validate outputs with data owners.

6. Deploy change to production only after sign■off.

---

8. Troubleshooting & FAQs

8.1 Common Login / Access Issues

• **Can’t log in / “Hmm, that didn’t work”**

→ Check username/password, Caps Lock, try again. If still failing, reset password or contact admin.

• **“Account locked”**

→ Too many failed attempts. Wait 15 minutes and contact admin if still locked.

• **I don’t see certain modules in the sidebar**

→ Your role may not have access; ask your admin to enable modules for you via **Security/Admin**.

8.2 When Jobs Don’t Run as Expected

• **Job did not run at scheduled time**

- Verify the **scheduler service** is running.

- Check **Jobs** to ensure schedule is enabled.

- Check **Job Status & Logs** for missed/failed runs.

- **Job shows success but data not updated**

- Verify target connection and schema.

- Check that mapping logic aligns with expectations.

- Confirm date ranges and filter conditions.

8.3 Data Mismatch / Wrong Numbers

- Compare:

- Source data using **Manage SQL Test** or DB client.

- Target data using a SQL query.

- Mapping logic to confirm each field.

- Use **Job logs** to:

- Check row counts.

- Identify dropped/filtered rows.

8.4 Who to Contact

In your organization, typical roles are:

- **DMS Administrator** – for:

- Access and roles.

- Module enable/disable.

- License and environment issues.

- **Data Engineer / ETL Developer** – for:

- Complex mapping logic.

- Database performance issues.

- New source system onboarding.

- **Business/Data Owner** – for:

- Confirmation of business rules.

- Validation of numbers and results.

You can now use this guide as a “companion manual” while working through the UI.

For any screen, ask yourself:

- **What is the role of this module in the overall data flow?**
- **Where is the data coming from and where is it going?**
- **How does this configuration affect the jobs and the final reports/dashboards?**

If you keep those questions in mind, the DMS Tool will feel like a **structured, end■to■end pipeline builder**, not just a set of separate screens.