

| Business Template  **DWH Testing** |
| --- |
| 0.PREREQUISITE **1. Via PgAdmin SQL editor invoked by `Tools->Query Tool` OR DBeaver execute following scripts  2. For newly created dwh\_src\_hw\_db database execute 1\_create\_sources\_postgres.sql file**    **3. Check that you have:**    **4. For newly created dwh\_hw\_db database execute 2\_create\_landing\_postgres.sql, 3\_create\_DWH\_layer\_postgres.sql and 4\_create\_DM\_layer\_postgres.sql files**    **5. Check that you have:**     1. DATA RECONCILIATION TASK **As you already know, Data Reconciliation is the process of verifying that data from different systems or layers is complete and consistent. In our case, we have a Source System Database (dwh\_src\_hw\_db) and a Target System Database (dwh\_hw\_db), each with its own layers and entities.**  **Since data is typically uploaded from SRC → LND without any transformations, our goal is to reconcile and ensure that all data from SRC is fully represented in LND with the last version of data. There are several approaches to achieving this, but we will focus on two simple methods that do not require additional tools: Using SQL and Using Python**  **1. Reconciliation with SQL**  **As the dwh\_hw\_db database serves as our target DWH, we need to connect it with the external source database (dwh\_src\_hw\_db). To enable this, we use dblink and first, we need to create the extension (make sure you do this in the target database):**    **After we have that extension we can connect to another db with use of dedicated user and password:**    **And now, our goal is to compare rows between dedicated tables of 2 DBs. For this example, let’s check: s1.s1\_channels (from the Source DB) vs. lnd.lnd\_s1\_channels (from the Landing layer in Target DB) with use of FULL OUTER JOIN:**    **And the output is:**    **But even though the task seems almost complete, the raw output is not very readable or convenient for quickly identifying mismatches.  To improve this, we can wrap the query in a result CTE and classify rows by clear reconciliation statuses, making it easier to spot differences.**  **And now, by ordering the output by reconciliation\_status, we can clearly see which rows require our attention first:**      **But still, in the reconciliation output we can see both mismatched and correct rows together. To make the results more useful, it’s better to store only problematic cases in a dedicated results table.**  **Your task is:**   1. **Create a reconciliation\_results table in the lnd schema with the following structure:**    * **table\_name**    * **key\_column**    * **src\_id**    * **trg\_id**    * **reconciliation\_status**      1. **Perform reconciliation checks for all other tables using dblink (similar to what was done for s1.s1\_channels vs. lnd.lnd\_s1\_channels).** 2. **Populate the reconciliation\_results table with rows that have only these statuses:**    * **'Only in source'**    * **'Only in target'**    * **'Mismatch in <column\_name>'**     ***(you can see the code in my sql file 5\_reconciliation\_with\_SQL.sql, additionally for critical path testing purposes I implemented reconciliation on each layer of the DWH)***  **2. Reconciliation with Python**  **The general logic for this task remains the same, but here we take another approach, which provides better flexibility, performance and security.**  **So to perform reconcile activities we need to connect to our DB with use of Python, but first lets set up proper environment for that – we suggest using of Jupyter NOtrebok. To set it up just:**  ***To mention that I already had Ananconda installed on my laptop***  **1. Go to**[**anaconda.com**](http://anaconda.com)    **2. Download Distribution Installer for your system**    **3. After you installed it run JupyterLab**    **4. To complete the homework task, upload the provided Python\_reconciliation.ipynb notebook and follow the instructions inside.**  **The detailed description of your assignment is also included in the file.**  ***(the code is in file Python\_reconciliation.ipynb )*** 2. DWH TESTING AND BUGS CREATION **1. DWH should be tested based on the provided DWH\_schema.pdf and DWH\_schema\_description.docx (same as in Data Mapping task) and test cases from previous HW.**    ***Based on the schema provided, in the smoke testing section, I tested the following aspects of the data pipeline:***   1. **Row-level integrity – Verified that each table contains data and is not empty.** 2. **Uniqueness of keys – Checked that primary and composite keys are unique, preventing duplicate records.** 3. **Null values in key columns – Ensured that key columns (IDs) do not contain nulls.** 4. **Schema validation – Confirmed that tables contain the expected columns as defined in the schema, with no missing or extra columns.** 5. **Referential integrity – Verified that foreign key relationships are valid (e.g., client\_id in sales exists in clients table).** 6. **Value sanity checks – Ensured numeric fields like quantities, units, and product costs are positive and dates are logically consistent (e.g., sale dates not in the future).** 7. **Data type and length checks – Confirmed that columns have the correct data types and character/numeric lengths as defined in the schema.**   **These tests collectively ensure that the data is complete, consistent, accurate, and conforms to the expected schema across all layers of the pipeline (S1, S2, landning, DWH, and DM).**  ***(the code is in files 7\_smoke\_tests\_source.sql and 6\_smoke\_tests.sql)***  ***Based on the schema provided, I performed reconciliation-based testing(critical path ) from source to landing (S1, S2) to DWH and finally to the dashboard (DM). Specifically, I tested the following:***   1. **S1 and S2 to Landing Layers**     * **Verified that all key tables (s1\_channels, s1\_clients, s1\_products, s1\_sales, s2\_channels, s2\_locations, s2\_clients, s2\_client\_sales) contain consistent records.**    * **Checked for:**       + **Rows missing in landing or source (Only in landing / Only in source)**      + **Column-level mismatches (e.g., channel\_name, location\_name, units, product\_cost, first\_purchase)**      + **Correct mapping of composite keys for sales tables (client\_id | channel\_id | product\_id | sale\_date)** 2. **Landing to DWH Layer**     * **Ensured that DWH tables (dwh\_channels, dwh\_locations, dwh\_clients, dwh\_products, dwh\_sales) accurately reflect the landing layer.**    * **Validated:**       + **Surrogate key mapping from landing to DWH**      + **Column-level consistency for names, costs, dates, and quantities**      + **Presence of all landing records in DWH** 3. **DWH to Dashboard Layer (DM)**     * **Checked that dashboard data (dm\_main\_dashboard) is consistent with the DWH source.**    * **Verified:**       + **All sales exist in the dashboard**      + **Correct aggregation of client, channel, location, product, and total cost information**      + **Column-level consistency for all relevant fields** 4. **General Reconciliation Approach**     * **Full outer joins between source and target at each stage**    * **Identification of mismatches vs matches**    * **Captured results in lnd.reconciliation\_results for tracking and reporting**    * **Generated summary counts and sample mismatched records**   ***(the code is in file 5\_reconciliation\_with\_SQL.sql )***  ***Based on the schema provided, I performed extended path testing to capture edge cases and data quality issues beyond the main reconciliation flow. Specifically, I tested the following:***   1. **S1 Clients – Middle Name**     * **Checked for NULL or empty middle\_name values in s1\_clients.**    * **Flagged records where middle\_name was missing as potential data quality issues.** 2. **S2 Clients – Valid From / Valid To Consistency**     * **Verified that valid\_from and valid\_to dates in s2\_clients match the DWH (dwh\_clients).**    * **Captured any mismatches between landing and DWH.** 3. **S1 Sales – Units Validation**     * **Identified records in s1\_sales where units <= 0.**    * **Ensured that all sales have positive units, marking invalid or suspicious entries.** 4. **S2 Client Sales – Sale Date Checks**     * **Flagged saled\_at as NULL or in the future in s2\_client\_sales.**    * **This ensures no future or missing sales dates propagate downstream.** 5. **S1 Products – Missing Attributes**     * **Checked for missing product\_name or cost in s1\_products.**    * **Ensured product records are complete before DWH ingestion.** 6. **S1 Channels – Location Mapping Edge Cases**     * **Verified that channel\_location from s1\_channels correctly maps to dwh\_locations.**    * **Captured any missing or mismatched location mappings.** 7. **Reporting**     * **Aggregated results in lnd.reconciliation\_results.**    * **Generated summary counts, sample source IDs, and target IDs for tracking and remediation.**   ***(the code is in file 8\_extended\_path.sql )***  **2. Found bugs should be created in JIRA. All bugs should have DWH\_ prefix. After creation of the bug please, move it to the In Progress column. Bug structure should be the following**   | **Summary** | **Short summary of the issue.** | | --- | --- | | **Description** | **Link to the Test case. In case if bug was found w/o test case – leave this field blank.**  **STR (Steps to reproduce)**  **ER (Expected Result)**  **AR (Actual Result)** | | **Environment**  **\*Will be the same for all bugs** | **Postgres version**  **Windows Version.**  **Other details that you will find useful.** | | **Attachments** | **Screenshot(s) with highlight of issue.** | | **Priority** | **Highest or**  **High or**  **Medium or**  **Lowest or**  **Low** | | **Severity** | **Critical or**  **Major or**  **Minor or**  **Trivial** | | **Assignee** | **Assign a ticket on yourself** |     **3. Don't add new test cases/adjust existing in the Test Rail.**  **4. If you will find bug w/o Test case - it's okay. No need to create new Test case. Just put info on how you found this bug in JIRA ticket.**  **5. Some issues in metadata shouldn't block your work for counts/duplicates/row-by-row comparison testing. Adjust SQL script for the actual table name/column name that we have in the DB. Put in the bug description that you change SQL script as a workaround.**  **6. Be very careful with critical/high priority/severity, if you will put such values please justify within bug why you think so.**  **7. If bug was found out using Test Case – please, put the link to the test case in the bug description**  **Here are the bugs I found. In addition, there may have been issues with column names that I haven’t listed here, as I had already corrected them when I began working with the database (e.g., channellocation, iast\_name).**  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1724**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1724)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1725**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1725)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1726**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1726)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1727**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1727)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1747**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1747)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1748**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1748)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1749**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1749)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1750**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1750)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1728**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1728)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1729**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1729)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1730**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1730)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1731**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1731)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1732**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1732)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1733**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1733)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1734**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1734)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1735**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1735)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1736**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1736)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1737**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1737)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1738**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1738)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1739**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1739)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1740**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1740)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1741**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1741)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1742**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1742)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1743**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1743)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1744**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1744)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1745**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1745)  [**https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1746**](https://dq-team-1-mimlefokko.atlassian.net/browse/SCRUM-1746) |