This hands-on assignment guide is designed to help you validate business rules, data transformations, and data quality using **Lakehouse SQL**, **Notebooks**, and **Lakehouse Explorer** — while leveraging **source-to-target mappings** and Fabric tools.

Hands-On Assignment Guide (Objective 2: SQL Test Case Development)

Platform: Microsoft Fabric
Target Skills: Data quality checks, transformation validation, SQL test case design, Lakehouse testing

---

✅ Step 1: Prepare the Lakehouse and Tables

- Use the **Lakehouse** created in Objective 1 (`TestLakehouse`).
- Ensure these tables exist:
  - `prod_sample_users` – Simulated source table
  - `synthetic_users_with_edge_cases` – Transformed target
- If not present, re-run Objective 1 scripts or recreate the tables manually.

✅ Step 2: Understand the Mapping Document

- Create or simulate a **Source-to-Target Mapping Document**:

Source Column	Target Column	Transformation Rule
`name`	`full_name`	As-is
`email`	`email_address`	As-is

`dob`	`birth_year`	Extract year from DOB
`id`	`user_id`	Remove negatives/nulls

- Store this in a OneNote page or Excel file inside the Fabric workspace.

✅ Step 3: Create Transformed Target Table Using SQL

Use **Lakehouse SQL Endpoint** or a SQL Notebook to run:

```sql
CREATE OR REPLACE TABLE transformed_users AS
SELECT
  id AS user_id,
  name AS full_name,
  email AS email_address,
  YEAR(dob) AS birth_year
FROM synthetic_users_with_edge_cases
WHERE id IS NOT NULL AND id >= 0;
```

✅ This simulates the transformation layer.

✅ Step 4: Write Test Case 1 – Record Count Validation

```sql
SELECT
  (SELECT COUNT(*) FROM synthetic_users_with_edge_cases
WHERE id IS NOT NULL AND id >= 0) AS expected,
  (SELECT COUNT(*) FROM transformed_users) AS actual;
```

✅ Compare expected vs. actual record count.

✅ Step 5: Write Test Case 2 – Null Check on Critical Columns

```sql
SELECT *
FROM transformed_users
WHERE user_id IS NULL OR full_name IS NULL OR
email_address IS NULL;
```

✅ Ensure critical fields are not null.

✅ Step 6: Write Test Case 3 – Duplicate Check

```sql
SELECT user_id, COUNT(*)
FROM transformed_users
GROUP BY user_id
HAVING COUNT(*) > 1;
```

✅ No duplicate `user_id` should exist.

✅ Step 7: Write Test Case 4 – Transformation Rule Validation

```sql
SELECT dob, birth_year
FROM synthetic_users_with_edge_cases s
JOIN transformed_users t ON s.id = t.user_id
WHERE YEAR(s.dob) <> t.birth_year;
```

✅ Ensure DOB transformation to birth year is applied correctly.

✅ Step 8: Write Test Case 5 – Value Range Check

```sql
SELECT *
FROM transformed_users
WHERE birth_year < 1900 OR birth_year >
YEAR(CURRENT_DATE);
```

✅ Birth year should be in a valid range.

✅ Step 9: Create a Test Results Table.

```sql
CREATE OR REPLACE TABLE test_results (
  test_case_id INT,
  test_description STRING,
  status STRING,
  failure_count INT
);
```

✅ Step 10: Insert Test Case Results (Example for Count Check)

```sql
INSERT INTO test_results
SELECT
  1 AS test_case_id,
  'Record count validation',
  CASE WHEN
    (SELECT COUNT(*) FROM synthetic_users_with_edge_cases
WHERE id IS NOT NULL AND id >= 0)
    =
    (SELECT COUNT(*) FROM transformed_users)
  THEN 'PASS' ELSE 'FAIL' END,
  ABS(
    (SELECT COUNT(*) FROM synthetic_users_with_edge_cases
WHERE id IS NOT NULL AND id >= 0) -
    (SELECT COUNT(*) FROM transformed_users)
  ) AS failure_count;
```

✅ Repeat similar inserts for other test cases.

✅ Step 11: View Consolidated Test Results

```sql
SELECT * FROM test_results ORDER BY test_case_id;
```

✅ Easy summary of all validations.

✅ Step 12: Automate Tests in a SQL Notebook

- Create a new Notebook (e.g., `SQLTestAutomationNotebook`).
- Use `%sql` cells to execute all test case queries in sequence.
- Schedule using **Data Pipeline** if needed.

✅ Step 13: Track Coverage with a Test Matrix

- Use **Excel or Power BI** to track:
  - Fields tested
  - Rule applied
  - Pass/fail status
- Optional: Build a **Test Case Dashboard** in Power BI using `test_results`.

✅ Step 14: Use Git Integration to Version Control SQL

- Connect workspace to GitHub or Azure DevOps.
- Commit all notebooks and test cases under version control.

✅ Step 15: Peer Review & Final Validation

- Share the workspace or notebook with peers.
- Conduct review of:
  - Mapping adherence
  - Test completeness
  - SQL logic correctness
- Capture feedback and finalize the test suite.

🏁 Final Deliverables

Deliverable	Description
`transformed_users`	Final transformed target table
`test_results`	Table summarizing all test case outcomes
`SQLTestAutomationNotebook`	Executes all validations
Mapping Document	Excel/OneNote mapping of source to target
Test Matrix	Optional Excel/Power BI view of test coverage

Test Code Download Steps *>>*

To download the assignment pack, you'll need to be logged in to ChatGPT to run Python code directly here. Since that's not currently enabled, you can **run the following Python code yourself** locally to generate the ZIP file:

Python Script to Create the SQL Test Assignment Pack

```python
import os
from zipfile import ZipFile

# Create directory
base_dir = "sql_test_assignment_pack"
os.makedirs(base_dir, exist_ok=True)

# Sample Mapping Document
with open(os.path.join(base_dir, "mapping_document.csv"),
"w") as f:
    f.write("""Source Column,Target Column,Transformation
Rule
name,full_name,As-is
email,email_address,As-is
dob,birth_year,Extract year from DOB
id,user_id,Remove negatives/nulls
""")

# SQL Script
with open(os.path.join(base_dir, "sql_test_cases.sql"),
"w") as f:
    f.write("""-- Create Transformed Table
CREATE OR REPLACE TABLE transformed_users AS
SELECT id AS user_id, name AS full_name, email AS
email_address, YEAR(dob) AS birth_year
FROM synthetic_users_with_edge_cases
WHERE id IS NOT NULL AND id >= 0;

-- Test Case: Record Count Check
SELECT (SELECT COUNT(*) FROM
synthetic_users_with_edge_cases WHERE id IS NOT NULL AND
id >= 0) AS expected,
       (SELECT COUNT(*) FROM transformed_users) AS
actual;

-- Test Case: Null Check
SELECT * FROM transformed_users WHERE user_id IS NULL OR
full_name IS NULL OR email_address IS NULL;

-- Test Case: Duplicate Check
SELECT user_id, COUNT(*) FROM transformed_users GROUP BY
user_id HAVING COUNT(*) > 1;

-- Test Case: Transformation Validation
SELECT dob, birth_year FROM
synthetic_users_with_edge_cases s
JOIN transformed_users t ON s.id = t.user_id
WHERE YEAR(s.dob) <> t.birth_year;

-- Test Case: Range Check
SELECT * FROM transformed_users WHERE birth_year < 1900
OR birth_year > YEAR(CURRENT_DATE);

-- Create Test Results Table
CREATE OR REPLACE TABLE test_results (
  test_case_id INT,
  test_description STRING,
  status STRING,
  failure_count INT
);
""")

# Notebook Template
with open(os.path.join(base_dir,
"sql_test_notebook_template.txt"), "w") as f:
    f.write("""# SQL Test Automation Notebook

## Run Transformation
%sql
-- Run table creation SQL

## Run Test Cases
%sql
-- Paste test queries here

## View Test Results
%sql
SELECT * FROM test_results;
""")

# Test Matrix
with open(os.path.join(base_dir, "test_matrix.csv"), "w")
as f:
    f.write("""Test Case ID,Description,Source
Column,Target Column,Rule,Tested,Status
1,Record Count Check,id,user_id,Remove
nulls/negatives,Yes,Pending
2,Null Check,all fields,all fields,No nulls,Yes,Pending
3,Duplicate Check,user_id,user_id,Unique
constraint,Yes,Pending
4,DOB to Birth Year,dob,birth_year,YEAR(dob),Yes,Pending
5,Birth Year Range,dob,birth_year,Valid range
check,Yes,Pending
""")

# Create ZIP
with ZipFile("fabric_sql_test_assignment_pack.zip", "w")
as zipf:
    for root, _, files in os.walk(base_dir):
        for file in files:
                zipf.write(os.path.join(root, file),
arcname=file)
```

Once you run this, you'll get a ZIP file named `fabric_sql_test_assignment_pack.zip` containing:

- `mapping_document.csv`
- `sql_test_cases.sql`
- `sql_test_notebook_template.txt`
- `test_matrix.csv`