Below is a **15-step hands-on assignment guide for Objective 3: Automation of SQL Tests** using the **Microsoft Fabric** platform.

This guide will help you automate SQL test execution using **Notebooks**, **Spark**, **Lakehouse SQL**, and **Data Pipelines**, and optionally integrate it into **CI/CD with Git**. You'll implement automation that validates your data transformations and ensures results are testable and repeatable.

Hands-On Assignment Guide (Objective 3: Automate SQL Tests)

Platform: Microsoft Fabric
Target Skills: Test automation using Notebooks & Pipelines, SQL integration, CI/CD, DevOps

---

✅ Step 1: Ensure Objective 2 Outputs Are Ready

Make sure the following from Objective 2 are in place:

- `transformed_users` table
- `test_results` table
- `SQLTestAutomationNotebook`
- `mapping_document.csv` and `test_matrix.csv` (optional for coverage tracking)

✅ Step 2: Open or Create a New SQL Test Automation Notebook

- Go to your Lakehouse.
- Create a new **Notebook** (or reuse `SQLTestAutomationNotebook`).
- Attach it to the **same Lakehouse** (e.g., `TestLakehouse`).

- Choose **PySpark** as the default language (you can still run `%sql` blocks).

✅ Step 3: Add `%sql` Blocks for Each Test Case

Use `%sql` magic to add the test logic you wrote in Objective 2:

```sql
%sql
-- Test Case 1: Record Count Check
INSERT INTO test_results
SELECT 1, 'Record Count Check',
       CASE WHEN (SELECT COUNT(*) FROM transformed_users)
=
                   (SELECT COUNT(*) FROM
synthetic_users_with_edge_cases WHERE id IS NOT NULL AND
id >= 0)
            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));
```

Repeat similar blocks for all test cases.

✅ Step 4: Create a Clear Section for Each Test Case

Structure your notebook as:

- **Section 1**: Setup / Initialization
- **Section 2**: Transformation Execution (re-create `transformed_users`)
- **Section 3**: Individual Test Cases
- **Section 4**: Insert Results into `test_results`
- **Section 5**: Final Summary View

✅ Step 5: Create a SQL Summary Report Block

Add this at the end of your notebook:

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

✅ View the result directly in the notebook output pane.

✅ Step 6: Save and Name the Notebook Clearly

Name the notebook something like:

```
Automated_Data_Validation_Tests
```

✅ Step 7: Create a Data Pipeline

- Go to **Data Pipelines** > **New Pipeline**.
- Drag in a **Notebook activity** and attach your automation notebook.

✅ Step 8: Add Trigger to Schedule Execution

- Set up a **Trigger**:
    - Run **daily**, **after ingestion**, or **on-demand**.
    - You can also set parameters if needed later.

✅ Step 9: Publish the Pipeline

- Click **Publish All**.
- Give the pipeline a meaningful name: `Nightly_SQL_Test_Validation`.

✅ Step 10: Run the Pipeline and Monitor

- Trigger the pipeline manually.
- Check the output log.
- Go back to the Lakehouse → Tables → `test_results` to view updates.

✅ Step 11: Visualize Test Results in Power BI (Optional)

- In Power BI (within Fabric):
  - Connect to your Lakehouse.
  - Use the `test_results` table.
  - Create a summary report showing:
    - Pass/Fail counts
    - Trends over time (if you timestamp results)

✅ Step 12: Enable Git Integration (Optional CI/CD Step)

- Go to Workspace settings → **Git integration**.
- Connect to a **GitHub** or **Azure DevOps** repo.
- Commit:
  - Your notebooks
  - SQL scripts
  - Test documentation

✅ Step 13: Create a Dev → Test → Prod Flow (Optional)

- Duplicate your workspace for Dev, Test, and Prod.
- Promote test cases via pull requests in Git.
- Automate deployment using Azure DevOps pipelines or GitHub Actions.

✅ Step 14: Build an Error Notification System (Advanced Optional)

- In your notebook or pipeline:
  - Add logic to raise errors if any test fails.
  - Send email alerts via Power Automate or trigger an HTTP call to Teams/Slack.

✅ Step 15: Document the Automation Workflow

In OneNote or Markdown:

- Describe:
  - Purpose of each test
  - How the notebook works
  - Pipeline schedule
  - Links to GitHub repo
- Include screenshots of success/failure and Power BI dashboards.

🏁 Final Deliverables

Item	Description
Notebook	Automated SQL test execution
Data Pipeline	Scheduled execution engine
test_results table	Central log of test outcomes

Power BI report *(optional)*	Visual QA report
Git integration *(optional)*	Version-controlled pipeline and notebooks

---

To download the automation ZIP pack directly from ChatGPT, you'll need to **log in to ChatGPT with Python access enabled** (available to Pro or Team users). Since that's not currently available in your session, here's a Python script you can **run locally** to generate the ZIP file:

🧾 Run This Python Script Locally

```python
import os
from zipfile import ZipFile

# Create directory for automation pack
base_dir = "sql_test_automation_pack"
os.makedirs(base_dir, exist_ok=True)

# 1. SQL Test Automation Notebook Template
notebook_template = """# Automated SQL Test Notebook - Microsoft Fabric

## Initialization
%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;

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

## Record Count Validation
%sql
INSERT INTO test_results
SELECT 1, 'Record Count Check',
       CASE WHEN (SELECT COUNT(*) FROM transformed_users)
=
                   (SELECT COUNT(*) FROM
synthetic_users_with_edge_cases WHERE id IS NOT NULL AND
id >= 0)
            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));

## Null Check
%sql
INSERT INTO test_results
SELECT 2, 'Null Field Check',
       CASE WHEN EXISTS (
         SELECT 1 FROM transformed_users
         WHERE user_id IS NULL OR full_name IS NULL OR
email_address IS NULL
       ) THEN 'FAIL' ELSE 'PASS' END,
       (SELECT COUNT(*) FROM transformed_users
        WHERE user_id IS NULL OR full_name IS NULL OR
email_address IS NULL);
```

```
## Transformation Validation
%sql
INSERT INTO test_results
SELECT 3, 'DOB to Birth Year Validation',
       CASE WHEN EXISTS (
         SELECT 1 FROM synthetic_users_with_edge_cases s
         JOIN transformed_users t ON s.id = t.user_id
         WHERE YEAR(s.dob) <> t.birth_year
       ) THEN 'FAIL' ELSE 'PASS' END,
       (SELECT COUNT(*) FROM
synthetic_users_with_edge_cases s
        JOIN transformed_users t ON s.id = t.user_id
        WHERE YEAR(s.dob) <> t.birth_year);

## Duplicate Check
%sql
INSERT INTO test_results
SELECT 4, 'Duplicate user_id Check',
       CASE WHEN EXISTS (
         SELECT user_id FROM transformed_users
         GROUP BY user_id HAVING COUNT(*) > 1
       ) THEN 'FAIL' ELSE 'PASS' END,
       (SELECT COUNT(*) FROM (
         SELECT user_id FROM transformed_users
         GROUP BY user_id HAVING COUNT(*) > 1
       ));

## View Results
%sql
SELECT * FROM test_results ORDER BY test_case_id;
"""
with open(os.path.join(base_dir,
"sql_test_automation_notebook_template.txt"), "w") as f:
    f.write(notebook_template)
```

```python
# 2. README Guide
readme = """# SQL Test Automation Guide - Microsoft
Fabric

Includes:
- SQL automation notebook
- Test result insertion queries
- Pipeline configuration concept
- Optional CI/CD and alerting logic

Flow:
1. Prepare Lakehouse with test tables
2. Run notebook manually
3. Link to a Data Pipeline
4. Optionally set Git CI/CD
5. Use Power BI to visualize results
"""
with open(os.path.join(base_dir,
"README_Automation_Guide.txt"), "w") as f:
    f.write(readme)

# 3. Pipeline Diagram (Text Format)
diagram = """# Pipeline Flow

[Start]
   |
[Run SQL Automation Notebook]
   |
[Write to test_results Table]
   |
[Optional: Send Alerts or Dashboard Refresh]
   |
[End / Scheduled Trigger]
"""
with open(os.path.join(base_dir, "pipeline_diagram.txt"),
"w") as f:
    f.write(diagram)

# Create ZIP
with ZipFile("fabric_sql_test_automation_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)

print("ZIP file 'fabric_sql_test_automation_pack.zip'
created successfully!")
```