QA Training Guide (TDD & QA) AZURE/ AWS Track – Intermediate

***Author:*** *Marusuwan, Puritat & Sayanwisuttikam, Atipar*

***Role****: QA Competency leads*

***Email****:* [puritat.marusuwan@adastragrp.com](mailto:puritat.marusuwan@adastragrp.com) & [atipar.sayanwisuttikam@adastragrp.com](mailto:atipar.sayanwisuttikam@adastragrp.com)

***Date****: 22-Feb-23*

***Version****: 1.00*

# Overview

This document aims to train you for AZURE/ AWS QA. This document will provide/ walk you through QA for AZURE/ AWS cloud practices.

# What is QA?

QA testing is primarily used to verify that a piece of software is providing the same output as required by the end-user or business.

# QA in Agile projects

The Development Team is responsible for the work done in the Sprint. So, testing methods and approaches are part of the Development Team’s responsibilities. Regardless of members have a title as QA or not, in the end, the accountability goes to the whole team.

The team’s Definition of Done (DoD) sets the granularity of testing. However, the Development Team often defines the details without involving the Product Owner. That’s when the problems begin because it creates a mismatch of expectations between the Product Owner and Development Team. The DoD defines what “Done” means for the entire Scrum Team. So, all members should play a part in crafting the DoD.

# How the Training is Organized

## Training structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Activities | Prerequisite | Duration | Grading | Session |
|  | Installation   1. MySQL 2. Notepad++ or your preferred IDE | Completion of Level 1 QA training. Knowledge on basic and medium complex SQL. | 2-3 hours. | No grading. | This step does not require any session or recording. |
|  | ‘QA Session I’ will be conducted by Competency lead (Kula or Puritat or Atipar). | You can proceed to Step 2 without completion of Step 1. | 45 min. | No grading. | This activity will be recorded. |
|  | Assignment 1 (Creating test cases). | You cannot proceed to this step without Step 2 completion and graded. | 2.0 days | Candidate will be graded very carefully based on how they do the assignment/ task. | Feedback will be given after the task is completed |
|  | Assignment 2 (Load the data into QA) | You cannot proceed to this step without Step 3 completion and graded. | 2.0 days | Candidate will be graded very carefully based on how they do the assignment/ task. | Feedback will be given after the task is completed |
|  | ‘QA Session II’ will be conducted by Competency lead (Kula or Puritat or Atipar). | You can proceed to this step without completing Step 4. | 45 min. | No grading | This activity will be recorded. |
|  | Assignment 3 (SCD QA) | You can proceed to this step without completing Step 5. | 2.0 days | Candidate will be graded very carefully based on how they do the assignment/ task. | Feedback will be given after the task is completed |
|  | Assignment 1 (review of test cases). | You cannot proceed to this step without Step 3 completion. | Ad-hoc basis (30 min for each person). | Candidate will be requested to re-do the task if they are unable to meet standards (refer to appendix section). Will be graded very carefully on how the test cases has been created. | Feedback will be given after the task is completed. No session will be recorded. |
|  | Assignment 1 grading | Without completion of Step 3, there will no grades given. | Immediate | Immediate |  |
|  | Assignment 2 (review of QA test cases and execution) | You cannot proceed to this step without Step 4 completion. | Ad-hoc basis (30 min for each person). | Candidate will be requested to re-do the task if they are unable to meet standards (refer to appendix section). Will be graded very carefully on how the test cases has been created. |  |
|  | Assignment 2 grading | Without completion of Step 4, there will no grades given. | Immediate | Immediate |  |
|  | Assignment 2 (review of SCD QA test cases and Execution) | You cannot proceed to this step without Step 6 completion. |  | Candidate will be requested to re-do the task if they are unable to meet standards (refer to appendix section). Will be graded very carefully on how the test cases has been created. |  |
|  | Assignment 3 grading | Without completion of Step 6, there will no grades given. | Immediate | Immediate |  |

## Required Software Installation and Environment

This first step will help on the tools need for QA in ETL/ ELT.

* + - 1. Download and install [MySQL Community Edition](https://dev.mysql.com/downloads/file/?id=516927).
      2. Install Notepad or Notepad++ or any preferred IDE for editing SQLs.
      3. Once Assignment III is completed and graded, work with competency leads to get Azure/ AWS subscriptions.

# QA Training Stage I

## ETL Testing Approach

### ETL Process Overview

Text

Description automatically generated with medium confidence

### Types of ETL Projects

* 1. **Data Acquisition Projects:** Aim to load the new source data.
  + Changed or a completely new source of information.
  + It is required from the QA team to load and validate the new sources.
  + No changes in TSD or Mapping.
  + Watch out for impact to change capture when changes are made on existing sources.
  1. **Data Transformation Project:** Rely on already tested source data in order to create the target tables for the project.
  + No new sources of information as all the source data has already been tested.
  + New or changed target tables.
  + No/minimal changes in Extract specifications.
  1. **Data Extracts Projects:** Aim to deliver a solution which extracts data from the DB to files.
  + New or changed mapping document.
  + Changed or new Extract specifications document.
  1. **Mixed**

### QA for ETL Testing Approach

* + - * 1. **Preparation**

**Timelines**

It is important to know how much time you have for your work. Discuss the timelines with your QA team lead. Do refer to this document section 4.1.

**Access**

Database – Make sure you have the required access to the databases which are source or target of your process. You can’t test something you don’t see.

SharePoint – All documentation is stored on the team’s SharePoint and you will need to access it order to complete your tasks.

Tools – Install the necessary software for your work before you start testing. Sometimes it takes time, and it may delay your work.

* + - * 1. **Test Design**

**Description**

Main QA document, which contains all QA test cases, scripts and materials needed for the testing of a system.

**Content**

Test Cases - Spreadsheet with test cases and their validation scripts.

Supporting Scripts - Scripts which will build(load) the QA results.

Create QA table statements.

Load QA table statements.

Change Capture scripts.

**Usage**

Create this document with the QA test cases.

Create the validation scripts for each case and add them to the document.

After the execution step, update the results in the design.

After completion of the testing, you may have to review with project team.

**What is a Test Case? -** Each validation (case) which the QA wants to perform during the testing phase:

**Test Case ID and Name**: number or name of the current case

**Description**: What we want to test with this case.

**Expected results**: What we expect after the execution of the case (script)

**Actual results**: What is the actual result from the execution of the case (script)

**Scripts**: The script used for testing the case/

**Status**: Passed/Failed/Not executed.

Table

Description automatically generated

**Test Cases**

**Test Cases for Target Tables build criteria validation:**

Compare the count of the records in each ETL Target table with the count of the records in the corresponding Table with expected results.

Compare the count of records between corresponding Target table and Table with expected results where there is match on Primary Key values.

**Test Cases for Target Tables Primary Key values validation:** Having match in the number of records in the ETL Target tables with the number of records in the corresponding tables with Expected results, look for:

PK values available in the Expected results, but not found in the corresponding ETL Target tables.

PK values available in the ETL Target tables, but not found in the corresponding Tables with expected results.

List samples of such records and trace the records back to the source data and see if there are violated Transformation rules.

It is possible that there is a problem in the SQL that build the tables with the expected results. In this case fix it and run again the Build criteria validation and PK validation.

**Test cases for validating all other target table columns:**

Join each ETL Target table with the corresponding Table with expected results on the PK (matched in the previous steps) and validate each column value with the expected value. Count the records where there is no match and list sample of 10 or more records if available.

Analyze the cause for the difference and look for violated Transformation rules.

**Test case coverage:**

Match each test case to the corresponding Business Requirement(s) and look for not covered business requirements.

For each column in the target table, which is not part of the PK, there should be a test case created!!!

* + - * 1. **Build Criteria Script**

**Description:** Performs any operations which are necessary for the QA to load the QA target table with data. It is based on the rules found in the BA documents (joins, filters, unions).

**Create statement:** Based on the structure of the table, you will have to create an SQL script which creates the QA result table.

**Insert and Select:** Based on the mapping documents you will have to write a script which selects, transforms this data from the source tables, and then inserts it into the QA target table.

* + - * 1. **Test Execution**

**Entry Criteria:**

QA environment ready.

All Test Cases are ready, and all corresponding Test Scripts developed. Test Data prepared for all Test Cases.

ETL code ready for testing and uploaded in the Version Control System.

ETL code installation instruction and all required code documentation available.

**ETL code installation and execution:**

Get all code files from the Version Control System in the proper folders in the QA environment.

Follow exactly all installation instructions and verify that there are no issues with the installation.

Using the prepared Test Data execute all ETL code modules in the provided by the documentation sequence and

Verify that there are no issues with the execution.

Any exceptions and error log messages are escalated as defects.

**Test Cases execution and Test Results validation:**

Load all source data:

Load all source files using available from the DB tools. When possible, use also available ETL tools.

It is good to load all source columns with the appropriate types, but sometimes is possible to load all as VARCHAR/CHAR and cast later.

Copy all Target Tables populated from the ETL execution into separate set of tables. This will avoid conflicts when someone else from the QA team is using the same environment for other QA activities.

Based on the Transformation specification build the tables with expected data.

Run all Test Cases based on your Test Design document and validate all results.

Analyze each discrepancy found in the results and if the cause for the discrepancy is a code problem open a Defect.

Fill in Actual Results and Results in the Test Design, based on the execution results.

* + - * 1. **Results & Defects**

**Results**

The QA will review the results with their QA team lead.

Based on the results from the execution of all scripts, the QA will inform the DEV team of any discrepancies.

For each discrepancy (failed test) the QA will raise a defect in the Defect Tracking System.

**Defects**

The tester will record a defect in the Defect Tracking System. Full details of the defect should be recorded and evidence of the problem (i.e., screen prints) saved as attachments. Each defect starts with the status of NEW and is assigned to the Test Lead. All defects are discussed on Daily status meetings.

The Test Lead will change the status from “NEW” to “OPEN”, “REJECTED”, “MONITORED”, or “DEFERRED” based on the decision made at the daily meeting. OPEN variances will be assigned to the developer as confirmed by the Technical Lead, by the variance coordinator.

The developer will only work on variances that have a status of OPEN which are assigned to them.

After fixing the problem, the developer will change the status to FIXED, assign it back to the development team lead, and record the resolution and root cause.

The test lead and technical lead will determine when the fix can be implemented, and when the variance will be ready for retest. This will be discussed at the daily status meeting, and assigned back to the tester who detected it.

The tester will retest and change the status from FIXED to either CLOSED or RE-OPEN. If a variance is RE-OPENED, the tester should assign it to the system developer for handling as in step 3 above.

## Assignment I

You are a QA to an ETL project with given data. You will be performing the activities below:

1. Create a Test Design Document with the following content:
   1. Test cases for IDV load of QA table CUSTOMER.
   2. Test cases for ORG load of QA table CUSTOMER.
2. **Please refer to the Mapping document when creating the test cases.**
3. Refer to the Test Cases template in [S336](https://adastrabiz.sharepoint.com/:x:/r/sites/AdastraTH/LLT%20Repo/S336-QA%20Test%20Plan/Test%20Cases%20V01.00.xlsx?d=w9578e006346747209e2c997447c4e2da&csf=1&web=1&e=OZ0udv). Please use one sheet for **IDV**test cases and another one for **ORG**.
4. Submit the Test Case Document that you have created. You will receive the feedback how your test cases are written.



# QA Training Stage II

**NOTE:** Please complete QA Training Stage I before you do this step (do not skip).

## Validations

### Validation Steps

### Build Script: Approach

1. Start with the select statement:
   1. Add the target column with no rule.
   2. Add the transformation logic for this column.
   3. Do column by column, leave complex rules for last.
   4. Add comments on the same line if you have any doubts.
   5. Do no more than 5 columns at a time and test the select after every 5 or so columns.
2. Add the source tables and joins.
3. Add left joins for lookups.
4. In case of many joins or filters, split it into steps by using WITH.
5. Add transformation rules.
6. After successful run of the select add the insert statement before it and insert the data.
7. Fix any issues with columns’ size.

### Validation Scripts

1. **Records count** – Establishes that the number of records loaded by the DEV solution is the same as the number of records the QA expects.
   1. **Purpose:** Script used by the QA to validate, whether the number of records loaded from the QA build script, is the same as the number of records loaded from the DEV solution.
   2. **Possible Results**

|  |  |  |
| --- | --- | --- |
| **Case** | **Result** | **Reason** |
| DEV records > 0 and DEV records = QA records | Pass | Data is matching |
| DEV records=0 | Fail | No data was loaded. |
| DEV records > 0 and DEV records <> QA records | Fail | DEV and QA process do not have the same records count |

* 1. **Validation Scripts**

|  |
| --- |
| WITH **QA** AS (  SELECT COUNT(\*) C FROM **TXN\_QA**  )  , **DEV** AS (  SELECT COUNT(\*) C FROM **TXN\_DEV**  )  SELECT  ‘TC\_TXN\_1’ AS TEST\_NAME  , QA.C AS QA\_RECORDS  , DEV.C AS DEV\_RECORDS  , CASE  WHEN DEV.C >0 AND DEV.C=QA.C  THEN ‘**Pass**’  ELSE ‘**Fail**’  END AS **RESULT**  FROM **QA,DEV**; |

1. **PK Validation** – Establishes that the exact records (PK) which should be loaded by the DEV solution are loaded into the DEV table.
   1. **Purpose**
      1. Validates the number of matched PK between QA and DEV tables is the same as the number of records.
      2. Shows the PK-s which are not matching between QA and DEV tables.
   2. **Possible Results:**

|  |  |  |
| --- | --- | --- |
| **Case** | **Result** | **Reason** |
| 1.The number of matching PK = Number of DEV records AND  2. There is no PK (0 #) which is not matched | Pass | All keys are matching |
| The number of matching PK < Number of DEV records | Fail | Not all keys are matching |
| The number of matching PK > Number of DEV records | Fail | Duplicated PK or PK is not correct |

* 1. **Validation Scripts**

|  |
| --- |
| WITH DEV AS (  SELECT COUNT(\*) C FROM **TXN\_DEV**  )  , MATCH AS (  SELECT COUNT(\*) C FROM **TXN\_DEV** DEV  INNER JOIN **TXN\_QA** QA **ON DEV.TXN\_ID= QA.TXN\_ID**  )  , NOT\_MATCH AS (  SELECT COUNT(\*) C FROM TXN\_DEV DEV  FULL OUTER JOIN TXN\_QA QA  **ON DEV.TXN\_ID= QA.TXN\_ID**  **WHERE QA.TXN\_ID IS NULL OR DEV.TXN\_ID IS NULL**  )  **-- Step 1 Validates whether all PK are matched**  SELECT  'TC\_TXN\_2' AS TEST\_NAME  , DEV.C AS COUNT\_PK  , MATCH.C AS MATCHING\_PK  , CASE  WHEN DEV.C = MATCH.C AND NOT\_MATCHED.C =0  THEN 'PASS'  ELSE 'FAIL'  END AS RESULT  FROM DEV, MATCH, NOT\_MATCHED;  **-- Step 2 Prints the PK which are not matched**  SELECT  'TC\_TXN\_2' AS TEST\_NAME  , DEV.TXN\_ID AS DEV\_TXN\_ID  , QA.TXN\_ID AS QA\_TXN\_ID  FROM TXN\_QA QA  FULL OUTER JOIN TXN\_DEV DEV  ON DEV.TXN\_ID = QA.TXN\_ID  WHERE DEV.TXN\_ID IS NULL OR QA.TXN\_ID IS NULL; -- Either the QA PK is not found in the DEV table or vice versa |

1. **Columns Validation** – Establishes that the value in a column for a record is matching the same value for the same record between the DEV records and the QA records.
   1. **Purpose**: Validates that the DEV value for the column is the same as the QA value for the same column.
   2. **Possible Results**:

|  |  |  |
| --- | --- | --- |
| **Case** | **Result** | **Reason** |
| QA value for the column is matching DEV value for the column | Pass | Values are the same between QA and DEV |
| QA value for the column is not matching DEV value for the column | Fail | Actual results do not match expected results |

* 1. **Validation Scripts:**

|  |
| --- |
| -- Shows how many records have incorrect value for TXN\_TP between QA and DEV  **WITH**  MATCH AS (  **SELECT** COUNT (\*) C  **FROM** TXN\_QA  **INNER** **JOIN** TXN\_DEV  **ON** QA.TXN\_ID=DEV.TXN\_ID  **WHERE** COALESCE(QA.TXN\_TP,'-1') = COALESCE(DEV.TXN\_TP,'-1’))  , NOT\_MATCHED AS (  **SELECT** **COUNT** (\*) C  **FROM** TXN\_QA  **INNER** JOIN TXN\_DEV  **ON** QA.TXN\_ID=DEV.TXN\_ID  **WHERE** COALESCE(QA.TXN\_TP,'-1') <> COALESCE(DEV.TXN\_TP,'-1’))  , COUNT\_REC AS (**SELECT** COUNT (\*) C **FROM** TXN\_QA)  **SELECT**  'TC\_TXN\_3' AS TEST\_NAME  , COUNT\_REC.C AS TOTAL\_COUNT  , MATCH.C AS MATCHED\_RECORDS  , NOT\_MATCHED.C AS NOT\_MATCHED\_RECORDS  , **CASE**  **WHEN** MATCH.C=COUNT\_REC.C AND NOT\_MATCHED.C = 0  **THEN** 'PASS'  **ELSE** 'FAIL'  **END** AS RESULT  **FROM** MATCH, NOT\_MATCHED, COUNT\_REC; |

|  |
| --- |
| **--Show an example of 10 records where the value for TXN\_TP is not matching**  **SELECT**  QA.TXN\_ID AS PK  , QA.TXN\_TP as Expected\_Result  , DEV.TXN\_TP as Actual\_Result  **FROM** TXN\_QA QA  **INNER** **JOIN** TXN\_DEV **ON** QA.TXN\_ID=DEV.TXN\_ID  **WHERE** **COALESCE**(QA.TXN\_TP,'-1') <> **COALESCE**(DEV.TXN\_TP,'-1)  Limit 10; |

1. **Execution and Defects**
   1. **Steps:**
      1. Execute the scripts and save the results into log files (csv, txt, etc.)
      2. Based on the results from the execution fill in the test cases document in column actual results and status. The status update content can be referred to [LLT S428](https://adastrabiz.sharepoint.com/:w:/r/sites/AdastraTH/LLT%20Repo/S428-P3MO%20templates/Daily%20Status%20templates/Daily%20Status%20Template.docx?d=wde0e11ff90614820a628d1b4614ef74f&csf=1&web=1&e=rM2Dv2)
      3. Double check that the QA script is correct.
      4. Raise a defect for each discrepancy and put the differences within the defect.
      5. Inform the developer for the discrepancy.
   2. **Defect Resolutions:**
      1. Rerun the test where the defect was found.
      2. Check for mistakes on your side.
      3. Raise the defect in the Defect Tracking System (DTS)
      4. Notify the DEV via email with attachment(link) to the defect.
      5. Written form is always better.
      6. Protects you from misunderstanding.
      7. Assist DEV with any information he/she may require.
      8. Remind the DEV if no reply.
      9. Deploy new solution and retest.
         1. First the defect
         2. Then **all the cases** for this table
      10. If the defect is resolved notify the DEV and close the defect in the DTS
      11. Be Kind!
   3. **Email Contents for Defect Tracking:**

Use Template in [LLT S428](https://adastrabiz.sharepoint.com/:w:/r/sites/AdastraTH/_layouts/15/Doc.aspx?sourcedoc=%7BE6F3F6A8-22A1-4D7B-B414-BD228DA86714%7D&file=Defect%20Report%20v01.00.docx&action=default&mobileredirect=true) or the content should cover:

* + 1. Which build is being tested (latest?)
    2. Table
    3. Column/s
    4. Expected result
    5. Actual result
    6. Differences – Example of 10 records
    7. Mapping rule/TSD rule and document version.
    8. Any other comment
    9. Severity – Does it stop the process?
    10. Impact – Is it very important Table/Column?
    11. Approach the responsible DEV

1. **Automation of Validation Script**
   1. **Why automation:** The validation scripts are in the same template for every ETL process.
   2. **How:** Using replace script or a tool. Here is what you have to change:
      1. Records count: Change table’s name
      2. Primary Key: Change table’s name and PK.
      3. Column Validation: Change all below:
         1. Test Case name
         2. Table’s name
         3. Column’s name
         4. Primary key
         5. Column type(coalesce)

## Assignment II

You are a QA to an ETL project with given data. You will be performing the activities below:

1. Given the loading SQL script, in your database, execute the SQL Script source data loading.
2. Create validation script based on Test Case in Assignment I and Transformation Document given below.
3. Execute the validation scripts and record your results.
4. Submit the Test Case Document and the result accordingly. You will receive the feedback how your validation is executed and recorded.



# QA Training Stage III

**NOTE:** Please complete QA Training Stage II before you do this step (do not skip).

## Slowly Changing Dimension

### What is Slowly Changing Dimension (SCD)?

1. **Description**

Dimensions that change slowly over time, rather than changing on regular schedule, time-base. In Data Warehouse there is a need to track changes in dimension’s attributes in order to report historical data. In other words, implementing one of the SCD types should enable users assigning proper dimension's attribute value for given date. Example of such dimensions could be customer, product, employee, location, salesperson, etc.

1. **Importance**

Decision making

Reporting

Regulators

Maintenance

Data quality

Data Analysis

1. **Difficulties**

Data availability

Correct SCD should be used for a specific table

No Change Capture columns

### QA SCD Process Overview

In the Data Warehouse, historical data will be loaded periodically, depending on the frequency required. The exhibit below shows the example of daily data loading.

The QA Process will take place, for example from Day0 and Day1 as the figure below. The consecutive days of loading will also follow this principle.

### Testing Approach for SCD

1. **Build Scripts and Validation scripts**
   * + Prepared as usual based on the mapping documents, using the filter conditions for the first load.
     + Load the data into {QA\_TARGET\_TABLE}\_DAY0
2. **SCD Scripts preparation**
   * Change the script for DAY0, and load the data into a separate table, {QA\_TARGET\_TABLE}\_DAY1.
   * Prepare SCD script, based on PK and Change Capture columns.
   * Based on the data built in the table {QA\_TARGET\_TABLE}\_DAY0 and {QA\_TARGET\_TABLE}\_DAY1, load the data into {QA\_TARGET\_TABLE}
3. **Validation process**
   * Execute the DEV code with 1st data set for DAY0.
   * Execute Build scripts for DAY0 and validate the data for DAY0.
   * Execute the code with 2nd data set for DAY1.
   * Execute Build scripts along with SCD script for DAY1 and validate the data for DAY1.

### Changes of the Record

1. **New Records**
   1. DAY1 left join DAY0 on PK
   2. If DAY0 PK is NULL then the record is new
2. **Changed Records**
   1. Inner join DAY0 to DAY1 on PK
   2. If **any** of the non-PK columns from DAY1 is different from the same columns from DAY0, then the record is changed.
3. **Unchanged Records**
   1. Inner join DAY0 to DAY1 on PK
   2. If **all** the non-PK columns DAY0 are matching with their corresponding column in DAY1, then the record is not changed.
4. **Deleted**
   1. DAY0 left join DAY1 on PK
   2. If DAY1 PK is NULL then the record is deleted

### SCD Types

1. **SCD Type 0:** Transactional records. No change to the data. Every record is loaded as new record.

**Current Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **TXN\_ID** | **EFF\_DT** | **AMOUNT** | **TXN\_TP** |
| 1234566 | 2017-01-01 | 111.12 | DEP |
| 1234567 | 2017-01-01 | 1232132 | LN |
| 1234568 | 2017-01-01 | 990.02 | TP1 |

**New Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **TXN\_ID** | **EFF\_DT** | **AMOUNT** | **TXN\_TP** |
| 1234569 | 2017-01-02 | 250.01 | DEP |
| 1234570 | 2017-01-02 | 78.00 | LN |

**Final Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **TXN\_ID** | **EFF\_DT** | **AMOUNT** | **TXN\_TP** |
| 1234566 | 2017-01-01 | 111.12 | DEP |
| 1234567 | 2017-01-01 | 1232132 | LN |
| 1234568 | 2017-01-01 | 990.02 | TP1 |
| 1234569 | 2017-01-02 | 250.01 | DEP |
| 1234570 | 2017-01-02 | 78.00 | LN |

1. **SCD Type 1:** No history is kept. If the record is changed, it will be overwritten.
   * **Scenarios** New Data coming on DAY1
     + The record is new or changed, load the record from DAY1
     + If there is no new record, keep the existing record
   * **Steps**
     + Create DAY0 table with the current data in the target table
     + Create DAY1 table and load it with the new data using the build script
     + Empty the target table (QA)
     + Load the target table (QA) with the data from DAY0 and DAY1
     + Compare the QA table and the DEV table

**Current Data (DAY0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMER\_ID** | **EFF\_DT** | **END\_DT** | **CUST\_NM** | **CUST\_ADDR** |
| **199** | **2017-01-01** | **9999-12-31** | **Nikolay** | **Sofia** |
| **822** | **2017-01-01** | **9999-12-31** | **Lora** | **Varna** |
| **999** | **2017-01-01** | **9999-12-31** | **Stasi** | **Ruse** |

**New Records (DAY1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMER\_ID** | **EFF\_DT** | **END\_DT** | **CUST\_NM** | **CUST\_ADDR** |
| **199** | **2017-01-02** | **9999-12-31** | **Nikolay Popov** | **Varna** |
| **822** | **2017-01-02** | **9999-12-31** | **Lora** | **Varna** |

**Final Data after SCD**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMER\_ID** | **EFF\_DT** | **END\_DT** | **CUST\_NM** | **CUST\_ADDR** |
| **199** | **2017-01-02** | **9999-12-31** | **Nikolay Popov** | **Varna** |
| **822** | **2017-01-01** | **9999-12-31** | **Lora** | **Varna** |
| **999** | **2017-01-01** | **9999-12-31** | **Stasi** | **Ruse** |

**SCD Type 1 Scripts**

|  |
| --- |
| **CREATE** CUSTOMER\_DAY0 AS **SELECT** \* **FROM** CUSTOMER\_QA;  **TRUNCATE TABLE** CUSTOMER\_QA;  **INSERT** **INTO** CUSTOMER\_QA (CUSTOMER\_ID, EFF\_DT, END\_DT, CUST\_NM, CUST\_ADDR)  **-- Select unchanged records that exist in both DAY0 and DAY1 data OR records that are missing from the DAY1 data**  **SELECT**  DAY0.\*  **FROM** CUSTOMER\_DAY0 DAY0  **LEFT OUTER** **JOIN** CUSTOMER\_DAY1 DAY1  **ON** DAY0.CUSTOMER\_ID = DAY1.CUSTOMER\_ID  **WHERE** (DAY1.CUSTOMER\_ID **is not NULL**  **AND** DAY1.CUST\_NM = DAY0.CUST\_NM  **AND** DAY1.CUST\_ ADDR = DAY0.CUST\_ ADDR) -- ***unchanged records***  **OR** DAY1.CUSTOMER\_ID is NULL; *--* ***Records in DAY0 file but not in DAY1***  **UNION ALL**  **-- Select new or changed records and load it into the table**  **SELECT**  DAY1.\*  **FROM** CUSTOMER\_DAY1 DAY1  **LEFT OUTER** **JOIN** CUSTOMER\_DAY0 DAY0  **ON** DAY0.CUSTOMER\_ID = DAY1.CUSTOMER\_ID  **WHERE** (DAY0.CUSTOMER\_ID **is not NULL AND** *–* ***CUSTOMER exists in both datasets***  (DAY1.CUST\_NM <> DAY0.CUST\_NM **OR** DAY1.CUST\_ ADDR <> DAY0.CUST\_ ADDR)) ***-- and there is a change in one or all fields***  **OR** DAY0.CUSTOMER\_ID **is NULL**; *--* ***new record*** |

1. **SCD Type 2**: Historical data is kept. When a record is changed, the previously active record is expired with new END\_DT = New Record START\_DT, and the new record is inserted with END\_DT = 9999-12-31

* **Scenarios:** If the record is:
  + **New:** Insert the record
  + **Not Changed:** Keep the record
  + **Change:** Insert the new record and expire the old record
  + **Deleted:** Expire the old record
* **Steps**
  + Create DAY0 table with the current data in the target table
  + Create DAY1 table and load it with the new data using the build script
  + Delete the target table (QA)
  + Load the target table (QA) with the captured data changes between DAY0 and DAY1 data
  + Compare the QA table and the DEV table

**Current State (DAY0)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUST\_ID** | **EFF\_DT** | **END\_DT** | **CUST\_NM** | **CUST\_ADDR** |
| **199** | **2017-01-01** | **9999-12-31** | **Nikolay** | **Sofia** |
| **822** | **2017-01-01** | **9999-12-31** | **Lora** | **Varna** |
| **999** | **2017-01-01** | **9999-12-31** | **Stasi** | **Ruse** |

**New Data Load (DAY1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUST\_ID** | **EFF\_DT** | **END\_DT** | **CUST\_NM** | **CUST\_ADDR** |
| **199** | **2017-01-02** | **9999-12-31** | **Nikolay Popov** | **Varna** |
| **822** | **2017-01-02** | **9999-12-31** | **Lora** | **Varna** |

**Final Data After SCD (DAY1)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUST\_ID** | **EFF\_DT** | **END\_DT** | **CUST\_NM** | **CUST\_ADDR** |
| **199** | **2017-01-01** | **2017-01-02** | **Nikolay** | **Sofia** |
| **199** | **2017-01-02** | **9999-12-31** | **Nikolay Popov** | **Varna** |
| **822** | **2017-01-01** | **9999-12-31** | **Lora** | **Varna** |
| **999** | **2017-01-01** | **9999-12-31** | **Stasi** | **Ruse** |

**SCD Type 2 Scripts**

|  |
| --- |
| --DAY0.EFF\_DT = ‘2017-01-01’  --DAY1.EFF\_DT = ‘2017-01-02’  **CREATE** CUSTOMER\_DAY0 **AS SELECT \* FROM** CUSTOMER\_QA;  **DELETE FROM** CUSTOMER\_QA;  **INSERT INTO** CUSTOMER\_QA  (CUSTOMER\_ID, EFF\_DT, END\_DT, CUST\_NM, CUST\_ADDR)  **-- UNCHANGED records (present in DAY0 and DAY1)**  **SELECT** DAY0.\*  **FROM** CUSTOMER\_DAY0 DAY0  **INNER JOIN** CUSTOMER\_DAY1 DAY1  **ON** DAY0.CUSTOMER\_ID=DAY1.CUSTOMER\_ID  **WHERE** (**COALESCE(**DAY0.CUST\_NM**,'')=COALESCE(**DAY1.CUST\_NM**,‘’)**  **AND**  **COALESCE(**DAY0.CUST\_ADDR**,'')=COALESCE**(DAY1.CUST\_ADDR**,‘’) )** -- unchanged  **UNION ALL**  **-- NEW records. Records that exists in DAY1 but not in DAY0**  **SELECT** DAY1.\*  **FROM** CUSTOMER\_DAY0 DAY0  **RIGHT** **JOIN** CUSTOMER\_DAY1 DAY1  **ON** DAY0.CUSTOMER\_ID=DAY1.CUSTOMER\_ID  **WHERE** DAY0.CUSTOMER\_ID **IS NULL**  **UNION ALL**  **-- CHANGED old. Records that exist in both DAY0 and DAY1 and have changes in at least one non-PK column**  **SELECT**  DAY0.CUSTOMER\_ID,  DAY0.EFF\_DT,  DAY1.EFF\_DT - 1 **AS** END\_DT, ***-- Expires the old record***  DAY0.CUST\_NM,  DAY0.CUST\_ADDR  **FROM** CUSTOMER\_DAY0 DAY0  **INNER JOIN** CUSTOMER\_DAY1 DAY1  **ON** DAY0.CUSTOMER\_ID =DAY1.CUSTOMER\_ID  **WHERE** (**COALESCE**(DAY0.CUST\_NM,'') <> **COALESCE**(DAY1.CUST\_NM,'') **OR** **COALESCE**(DAY0.CUST\_ADDR,'')<>**COALESCE**(DAY1.CUST\_ADDR,'') )  **UNION ALL**  **-- CHANGED new. Records that exist in both DAY0 and DAY1 and have changes in at least one non-PK column**  **SELECT**  DAY1.CUSTOMER\_ID,  DAY1.EFF\_DT,  DAY1.END\_DT,  DAY1.CUST\_NM,  DAY1.CUST\_ADDR  **FROM** CUSTOMER\_DAY0 DAY0  **INNER** **JOIN** CUSTOMER\_DAY1 DAY1  **ON** DAY0.CUSTOMER\_ID = DAY1.CUSTOMER\_ID  **WHERE** (**COALESCE**(DAY0.CUST\_NM,'') <> **COALESCE**(DAY1.CUST\_NM,'') **OR** **COALESCE**(DAY0.CUST\_ADDR,'')<>**COALESCE**(DAY1.CUST\_ADDR,'') );  **UNION ALL**  **-- DELETED records (present in DAY0 and not in DAY1)**  **SELECT**  DAY0.CUSTOMER\_ID,  DAY0.EFF\_DT,  CAST(‘2017-01-02’ as DATE) - 1 **AS** END\_DT, ***-- Expires the old record***  DAY0.CUST\_NM,  DAY0.CUST\_ADDR  **FROM** CUSTOMER\_DAY0 DAY0  **LEFT JOIN** CUSTOMER\_DAY1 DAY1  **ON** DAY0.CUSTOMER\_ID=DAY1.CUSTOMER\_ID  **WHERE** (DAY1.CUSTOMER\_ID **IS NULL*) -- deleted*** |

1. **SCD Type 3:** Previous data is saved in a separate column and the new value is saved in the original column.
   * **Scenarios**
   * **The record is changed:** 
     + Keep the current record (DAY0)
   * **The record is changed:**
     + Update the column containing the current state of the data with the new value
     + Update the column containing the previous state of the data with the current value
2. **SCD Type 4:** Primary table contains only active records. History table contains all history records.
   * **Scenarios** If the record is:
     + - **New:** Load the record into the primary table
       - **Not Changed**: No change to either table
       - **Changed:**
         * Load the updated record into the primary table
         * Load the old record into the historical table with expired date
       - If the record is deleted – remove it from the primary table and expire it in the historical table

### Deletion Indicator

1. **Record is marked as Deleted (Soft Delete)** 
   * **Example**: When the customer has left the bank we still want to keep their information.
2. **How to determine record for deletion:**
   * Based on Requirements special processing or based on a record indicator.
   * The record exists on DAY0, but it is not available on DAY1
3. **What to do with such records**
   * Same as other columns. The current record changes its state for column DEL\_IND to ‘Y’

### Non-SCD Column

* + - 1. **Description:** Not all columns will trigger change capture process. If the only change is in a “Non SCD column”, this means that the record does not change, thus no change will be applied to the current record and no new record will be loaded. In the comparison between the current results and the new data, those records will be skipped in the verification for changes.
      2. **How to approach them:** Do not include them in your change capture scripts
      3. **How to identify them:** Usually the mapping will contain the information.

## Assignment III

You are a QA to an ETL project with given data. You will be performing the activities below:

1. Given the loading SQL script for SCD, in your database, execute the SQL Script source data loading.
2. Create validation script based on Test Case in Assignment I. Referring to Mapping Document and Transformation Document given below.
3. Execute the validation scripts and record your results.
4. Submit the Test Case Document and the result accordingly. You will receive the feedback how your validation is executed and recorded.



# Appendix

## Standard for writing test cases

* + - 1. Use the ATH test cases template for this assignment.
      2. Understand the template before starting your task.
      3. Before starting to write test cases, remember your scope and specification
      4. Write to the point clearly and precisely (must be readable).
      5. Try to put yourself in shoes of the clients/ customers/ stakeholders.
      6. Be granular while writing your steps for description, expected results, actual results and SQL scripts if any.
      7. A good test case should cover below based on the scenario.
         1. Structure validation 🡪 Unit test (TDD)
         2. Constraint validation if any
         3. Mapping validation (independent layers) 🡪 Unit test (TDD)
         4. Null validation 🡪 Unit test (TDD)
         5. Data consistency check
         6. Data validation check
         7. Data completeness check
         8. Data cleansing check if any
         9. Data transformation
         10. Duplicate validation
         11. Negative testing
         12. Data losses
         13. Source to target count/ mismatch/ duplicates/ incremental/ Transformation/ Regression and end to end testing.

## Template’s location in [LLT S428](https://adastrabiz.sharepoint.com/sites/AdastraTH/LLT%20Repo/Forms/AllItems.aspx?id=%2Fsites%2FAdastraTH%2FLLT%20Repo%2FS428%2DP3MO%20templates&viewid=8afe2425%2D9b0f%2D4e07%2D96b7%2D049940639aea)