# HOMEWORK 5

Test design artifacts task

1. Create a Test plan for this system.
2. Create a Test Strategy for this system. ([Page 9](#_Test_Strategy) of Test Plan)

# DATA WAREHOUSE TEST PLAN

[**DATA WAREHOUSE TEST PLAN**](#_2tyapzqutky6)1

[**Introduction**](#_e8ud2d2ebpd1)2

[Testing Goals and Verification Methods](#_yamqt2s9wn) 2

[Data Completeness](#_9u3xrfmf6l6w) 2

[Data transformation](#_3epzxdhgm9cj) 2

[Data Quality](#_lokcpbwe4qh5) 3

[Performance and Scalability](#_1f50ywg7811f) 3

[Integration testing](#_o33lb2mm9mu8) 3

[User-Acceptance Testing](#_nl27am95cr8) 4

[Regression Testing](#_mufill51u0yx) 4

[**Purpose**](#_c6ld3j1elusf)5

[**Components**](#_lob1n7omzyy1)5

[**Test Roles and Resources**](#_n86503nm0pvq)5

[**Items to be Tested**](#_xaknfhdzo8fh)6

[Tests from the Functional Area](#_idgtangsjt1d) 6

[Load Testing](#_8wxlz162ldbq) 7

[Security Testing](#_lftwb1p9zd3o) 7

[**Items not to be Tested**](#_qzkth291g4z7)7

[**Test Planning Readiness Assumptions and Needs**](#_pcm52z3bv191)8

[**Test Strategy**](#_dcfkdbxg51p9)9

[**Test Approach**](#_s9gsx5gutte6)10

[**Schedule**](#_pqllw7pthgjz)11

[**Deliverables**](#_cesj9b8epswy)11

[**Test Plan Approval**](#_abp1d3d08r2g)12

# Introduction

## Testing Goals and Verification Methods

Primary goals for DW verification over all testing phases include:

* Data completeness. Ensure that all expected data is loaded.
* Data transformation. Ensure that all data is transformed correctly according to business rules and/or design specifications.
* Data quality. Ensure that the ETL application substitutes default values, corrects or ignores and reports invalid data.
* Performance and scalability. Ensure that data loads and queries perform within expected time frames and that the technical architecture is scalable.
* Integration testing. Ensure that the ETL process functions well with other upstream and downstream processes.
* User-acceptance testing. Ensure the solution meets users' current expectations.
* Regression testing. Make sure that existing functionality remains the same every time you release new code.

## Data Completeness

One of the most basic will be to verify that all expected data loads into the data warehouse. This includes validation that all records, all fields and the full contents of each field are loaded. Strategies include:

* Comparing record counts between source data, data loaded to the warehouse and rejected records.
* Comparing unique values of key fields between source data and data loaded to the warehouse.
* Populating the full content of each field to validate that no truncation occurs at any step in the process.
* Testing the boundaries of each field to find any database limitations.

## Data transformation

Validating that data is transformed correctly based on business rules is a complex part of testing ETL with significant transformation logic. Following are some of the simple data movement techniques:

* Create a spreadsheet of scenarios of input data and expected results and validate these with business SMEs. This is a good requirements elicitation exercise during design and can also be used during testing.
* Create or identify test data that includes all scenarios.
* Utilize data profiling results to compare range and distribution of values in each field between source and target data.
* Validate correct processing of ETL-generated fields such as surrogate keys.
* Validate that data types in the warehouse are as specified in the design and/or the data model.
* Set up data scenarios that test referential integrity between tables.
* Validate parent-to-child relationships in the data.

## Data Quality

Data quality here is defined as how the ETL system handles data rejection, substitution, correction and notification without modifying data. To ensure success in testing data quality, include as many data scenarios as possible. Data quality rules will be defined during design, for example:

* Reject the record if a certain decimal field has nonnumeric data.
* Substitute null if a certain decimal field has nonnumeric data
* Validate and correct the state field if necessary, based on the ZIP code.
* Compare product code to values in a lookup table and if there is no match load anyway but report to users.

## Performance and Scalability

As the volume of data in the data warehouse grows, ETL load times can be expected to increase, and performance of queries can be expected to degrade. This can be mitigated by having a solid technical architecture and good ETL design. The aim of the performance testing is to point out any potential weaknesses in the ETL design, such as reading a file multiple times or creating unnecessary intermediate files. The following strategies will help discover performance issues:

* Load the database with peak expected production volumes to ensure that this volume of data can be loaded by the ETL process within the agreed-upon window.
* Compare these ETL loading times to loads performed with a smaller amount of data to anticipate scalability issues.
* Monitor the timing of the reject process and consider how large volumes of rejected data will be handled.
* Perform simple and multiple join queries to validate query performance on large database volumes.

## Integration testing

Usually, system testing only includes testing within the ETL application. The endpoints for system testing are the input and output of the ETL code being tested. Integration testing shows how the application fits into the overall flow of all upstream and downstream applications. When creating integration test scenarios, we will consider how the overall process can break and focus on touch points between applications rather than within one application. Consider how process failures at each step would be handled and how data would be recovered or deleted if necessary.

## User-Acceptance Testing

The main reason for building our data warehouse application is to make data available to business users. Users know the data best, and their participation in the testing effort is a key component to the success of a data warehouse implementation. User-acceptance testing (UAT) typically focuses on data loaded to the data warehouse and any views that have been created on top of the tables, not the mechanics of how the ETL application works. Consider the following strategies:

* Use data that is either from production or as near to production data as possible. Users typically find issues once they see the “real” data, sometimes leading to design changes.
* Test database views comparing view contents to what is expected.It is important that users sign off and clearly understand how the views are created.
* Plan for the system test team to support users during UAT. The users will likely have questions about how the data is populated and need to understand details of how the ETL works.
* Consider how the users would require the data loaded during UAT and negotiate how often the data will be refreshed.

## Regression Testing

Regression testing is revalidation of existing functionality with each new release of code and data. When building test cases, they will likely be executed multiple times as new releases are created due to defect fixes, enhancements or upstream systems changes. Building automation during system testing will make the process of regression testing much smoother. Test cases should be prioritized by risk in order to help determine which need to be rerun for each new release. A simple but effective and efficient strategy to retest basic functionality is to store source data sets and results from successful runs of the code and compare new test results with previous runs. When doing a regression test, it is much quicker to compare results to a previous execution than to do an entire data validation again.

# Purpose

The purpose of the test plan is to ensure that the, which includes a data conversion, is thoroughly tested, resulting in a successful implementation of new and existing functionality, business processes, reports, interfaces, and batch processes.

# Components

The functional are test plan consists of the following:

* Test Roles and Responsibilities
* Items to be Tested
* Test Strategy
* Test Approach
* Test Readiness Assumptions Met
* Deliverables
* Approval

# Test Roles and Resources

|  |  |
| --- | --- |
| **Role** | **Responsibilities - related to testing** |
| **Test Designers:**   1. Nazar, Data Quality, ETL Tester | * Develop and document test design for data conversion, functional, lifecycle, security, and performance testing * Create and document test cases based on test design * Communicate test updates to Test Coordinator and Functional Project Managers. |
| **Tester:**   1. Nazar, Data Quality, ETL Tester | * Run test cases during designated test periods * Document test results and problems * Work with developers to troubleshoot problems * Re-test problems fixes * Communicate test updates to Test Leads |
| **Project Manager:**   1. Project Manager | * Monitor and update project plan testing activities by functional area * Facilitate in the development of test plan, test design and test case for functional and lifecycle testing |
| **ETL Developer:**   1. Andrii 2. Ivan | * Perform unit test * Troubleshoot problems * Document problem updates in Test Director * Communicate unit testing progress to test lead * Work with Test Designers/testers on data validation planning and testing efforts |
| **DW Test Coordinator**  Nazar, Data Quality and ETL Tester | * Develop master test plan * Prepare test materials for functional areas use * Setup test and defect reporting * Setup Test Director for testing effort * Monitor test planning progress and execution efforts * Work with Functional Project Managers on test plan tasks * Keep functional areas and project team leads appraised of testing activities and results |

# Items to be Tested

Tests will be developed for the list of items below. Details of the items to be tested should be provided in the test design and scenario document - a future deliverable.

## Tests from the Functional Area

1. Data Validation
   1. Transformation values
   2. Set-up tables
   3. Row counts for each table
   4. Detailed review of specific cases (Transactions, Charges)
2. Business Processes (Including Business Requirements and Business Rules)
3. Reporting
4. SSIS Packages
5. Interfaces
6. Cross Functional Areas Processes (Lifecycle Testing)
7. Functional Area Key Process Performance (Timing Measurements)

## Load Testing

No load testing is currently planned.

## Security Testing

1. Double DUO identificatory login test
2. Users’ permissions and visibility test
3. Data access testing

# Items not to be Tested

* Platform back-end is out of scope as far as it has been already implemented and tested

# Test Planning Readiness Assumptions and Needs

A first level of testing and validation begins with the format acceptance of the logical data model and “low level design” documents including ETL design and functionality implemented in SSIS. All further testing and validation will be based on the understanding of each of the data elements in the data model and ETL design. For example, data elements and related fields that are created or modified through transformations or summary processes must be clearly identified and calculations for each of these data elements must be unambiguous and easily interpreted.

During the “low level design” reviews and updates, special consideration should be given to typical modeling scenarios that exist in the project. Examples follow:

1. Many-to-many attribute relationships are clarified and resolved.
2. Types of keys used are identified: surrogate keys versus natural keys.
3. Business analysts / DBA reviewed with ETL architects and developers.
4. All transformation rules, summarization rules, and matching and consolidation rules are documented.
5. ETL procedures are documented to monitor and control data extraction, transformation and loading.
6. Data consolidation of duplicate or merged data is properly handled.
7. Target data types, precision, field lengths are as specified in the design and/or the data model.
8. Mandatory (not null) indications are available for all fields
9. Default values are specified for fields where needed.
10. Acceptable values are provided for each field
11. Expected ranges for field values are specified where known.

# Test Strategy

Listed below are the various types of testing that will take place in four test phases.

|  |  |  |
| --- | --- | --- |
| **Testing** | **Role** | **Description** |
| Data validation | ETL Developer, Tester | Validating the data converted to MSSQL database. This is done early on in the project. |
| Unit | ETL Developers | Developer testing to validate that new code can be executed to completion with no errors. |
| Functional | Tester | To validate the application meets business requirements under various scenarios for a functional area. |
| Performance | Tester | Measuring the time an activity or key process takes to complete. This testing should be done throughout the test phases. |
| Load | Tester | Running key processes or activities under heavy usage simulation to determine if the application can perform adequately with high levels of activity/transactions |
| Regression | Tester | Regression testing to validate MSSQL patch and reg updates. |
| Acceptance | Tester | Customer testing of the application for approval for production. This testing typically is not in-depth testing. |
| Security | Key Security Contact, Tester | Testing security implemented. |
| Data Warehouse (DW) | DW Tester | Testing to validate changes to DW. This would involve DW Testing the DW data stage jobs in the conversion and the tester validating the repository queries. |
| Automated | Test Coordinator | Run redeveloped automated tests. |

# Test Approach

The four test phases are based on the development schedule for the database upgrade project along with the requirement to comply with financial aid regulation updates that need to be in place when the upgrade goes live.

**Phase 1:**

* Data Validation
* Performance
* Unit
* Functional
* Data Warehouse (internal testing within validating data stage jobs)

**Phase 2:**

* Cross-functional process
* Load
* Security
* Data Warehouse (repository testing and validation)

**Phase 3:**

* Regression testing

**Phase 4:**

* Business and client Acceptance

# 

# Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Activities** | **Role** | **Start Date** | **End Date** |
| Create Master Test Plan | Test Coordinator |  |  |
| Create Functional Test Plans | Test Coordinator |  |  |
| Create Test Design | Test Designer |  |  |
| Create Test Cases | Test Designer |  |  |
| Phase 1 Testing | Testers |  |  |
| Phase 2 Testing | Testers |  |  |
| Phase 3 Testing | Testers |  |  |
| Redevelop Automated Test Cases | Test Coordinator |  |  |
| Create Customer Acceptance Criteria | Test Coordinator |  |  |
| Phase 4 - Customer Acceptance Testing | Testers |  |  |

# Deliverables

* Functional Test Plan
* Functional Test Design
* Functional Test Cases

# Test Plan Approval

|  |  |
| --- | --- |
| **Functional Area Manager** | **Date Reviewed** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  [Type Functional Area Manager Here. Mgr signs above the line] |  |