**Test Data Management**

[Introduction 1](#_Toc1907147636)

[Purpose of the Document 1](#_Toc979204720)

[What is Test Data Management? 1](#_Toc964078269)

[Types of Test Data 3](#_Toc1282247903)

[Why Test Data Management? 4](#_Toc777392135)

[Test Data Management Techniques 5](#_Toc109727281)

[1. Data Masking 5](#_Toc2028569190)

[2. Data Subsetting 7](#_Toc127149680)

[3. Synthetic Data Generation 7](#_Toc1215502678)

## **Introduction**

In the world of software testing, **Test Data Management (TDM)** is a critical practice that ensures the availability, accuracy, security, and compliance of the data used during the testing process. The importance of test data cannot be overstated, as the quality of testing results directly depends on the quality and relevance of the data used in test cases. Test Data Management encompasses the processes and strategies for managing test data effectively, making it a key aspect of any successful software development lifecycle.

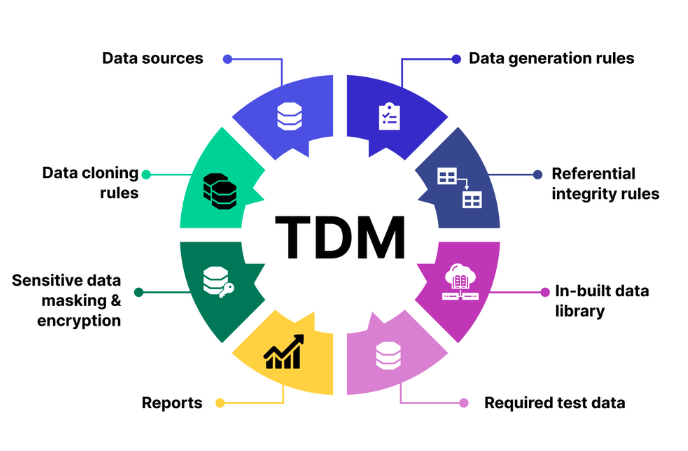
As applications become more complex, testing requires realistic and varied data sets to mimic real-world usage and uncover potential issues that might otherwise go undetected. Test data management helps ensure that testers have the appropriate data, whether for unit testing, integration testing, performance testing, or security testing.

### **Purpose of the Document**

This document explores the essential aspects of **Test Data Management** (TDM), including its definition, various types of test data, the tools that facilitate TDM, and the best practices to ensure efficient and secure test data handling. Additionally, it covers the importance of test data management in modern software testing and highlights various techniques used to manage test data.

## **What is Test Data Management?**

Test data management (TDM) is the process of planning, creating, and maintaining the datasets used in testing activities, ensuring that they are the right data for the right test case, in the right format, and available at the right time.  
Test data is the set of input values used during the testing process of an application (software, web, mobile application, API, etc). These values represent what a user would enter the system in a real-world scenario. Testers usually can write a test script to automatically and dynamically identify the right type of values to put into the system and see how it responds to those data.



For example, test data for the testing of a login page usually has 2 columns: a **Username** column and a **Password** column. A test script or automation testing tool can open the Login page, identify the Username field, the Password field, then input the values:

|  |  |
| --- | --- |
| **Username** | **Password** |
| user\_123 | Pass123! |
| testuser@email | Secret@321 |
| admin\_user | AdminPass# |
| jane\_doe | JaneDoePass |

You can have hundreds to thousands of such credential pairs representing unique test scenarios.   
  
But having a huge database does not immediately mean all of it is high-quality. There are 4 major criteria to evaluate test data quality, including:

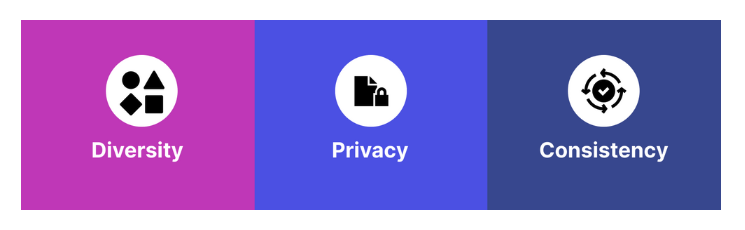
1. **Relevance:** it makes sense to have test data that accurately reflects the scenario being tested. Imagine testing the response of the login page when users enter the wrong set of credentials, but the test data being used is actually the correct, stored-in-database credentials. This results inaccurately.
2. **Availability:** what’s the point of having thousands of relevant data points yet you can’t retrieve them for testing activities? Usually, QA teams have clearly defined role-based access for test data, so TDM activities are also about assigning the right level of access to the right personnel.
3. **Updated:** software constantly changes, bringing with it new complexities and dependencies. The responsibility of QA teams is to be aware of those updates and make changes to the test data accordingly to ensure that results accurately reflect the current state of the software.
4. **Compliance:** aside from the technical aspects, we should never forget compliance requirements. QA teams sometimes leverage directly production data for testing activities due to its instant availability, but production data is a tricky domain: it may contain confidential information protected by GDPR, HIPAA, PCI, or other data privacy focused policies.

## **Types of Test Data**



1. **Positive Test Data:** This type of data consists of input values that are valid and within the expected range and is designed to test how the system behaves under expected and normal conditions. Examples: a set of valid usernames and passwords that allows users to login to their account page on an eCommerce site.
2. **Negative Test Data:** In contrast with positive data, negative test data consists of input values that are invalid, unexpected, or outside the specified range. It is designed to test how the system behaves when users do something out of the “correct” path intended. Examples: a set of usernames and passwords that is too long.
3. **Boundary Test Data: T**hese are values at the edges or boundaries of acceptable input ranges chosen to assess how the system handles inputs at the upper and lower limits of the allowed range.
4. **Invalid Test Data: T**hese are data that does not accurately represent the real-world scenarios or conditions that the software is expected to encounter. It does not conform to the expected format, structure, or rules within a given context.

## **Why Test Data Management?**



Here are some reasons why you should have your test data management process in place:

1. **Diversity:** high test coverage is synonymous with covering a rich array of test scenarios, and subsequently having test data for all of those scenarios. A simple registration page, for example, already requires so many datasets to cover all of the possible scenarios that can happen there:
   1. Valid credentials
   2. Empty username
   3. Empty password
   4. Incorrect username
   5. SQL injection attempt
   6. Special characters
   7. Too long username
   8. Too long password  
        
      There can be more, depending on the complexity of the registration page under test. for example, tends to establish multi-layer authentication for higher security, and that translates into more complex testing. TDM is there to ensure that test data for every test scenario is well-categorized and organized to best facilitate the actual testing activity.
2. **Data Privacy:** without good TDM practices, testers can risk using PII (personally identifiable information) to test, which is a breach in security. There are so many things you can do in TDM to prevent this from happening, such as data anonymization, which is essentially a process to replace real, sensitive data with similar but fictitious data. If teams decide to use real data, they can mask (i.e. encrypt) specific sensitive data fields and use only the most necessary. Several teams employ Dynamic Data Masking (DDM) to dynamically mask data fields based on user roles and permissions.
3. **Data Consistency:** QA teams also need to ensure that their test data is uniform across the entire systems, adhering to the same format and standards, and even the relationships among the datasets must be continuously maintained over time when the complexity of the system grows.

## **Test Data Management Techniques**

### **1. Data Masking**

Data masking is the technique used to protect sensitive information in non-production environments by replacing, encrypting, or otherwise “masking” confidential data while retaining the original data's format and functionality. Data masking creates a sanitized version of the data for testing and development purposes without exposing sensitive information.

The way data is masked depends on the algorithms QA teams chose. After cloning the data, there are quite a lot of ways to “play” with it and turn it into a completely new set of data in which the original identity of the users is protected. For example, we can:

|  |  |
| --- | --- |
| **Data Masking Technique** | **Definition + Examples** |
| **Substitution** | **Definition**: Replace actual sensitive data with fictional or anonymized values. You can leverage Generative AI for this approach; however, note that creating entirely new data is resource intensive.  **Example**: Replace actual names with randomly generated names (e.g., John Doe). |
| **Shuffling** | **Definition**: Randomly shuffle the order of data records to break associations between sensitive information and other data elements. This approach is faster and easier to achieve compared to Substitution.  **Example**: Shuffle the order of employee records, disconnecting salary information from individuals. |
| **Encryption** | **Definition**: Use encryption algorithms to transform sensitive data into unreadable ciphertext. Only authorized users with decryption keys can access the original data. This is a highly secure approach to take.  **Example**: Encrypt credit card numbers, rendering them unreadable without proper decryption. |
| **Tokenization** | **Definition**: Replace sensitive data with randomly generated tokens. Tokens map to the original data, allowing reversible access by authorized users.  **Example**: Replace social security numbers with unique tokens (e.g., Token123). |
| **Character Masking** | **Definition**: Mask specific characters within sensitive data, revealing only a portion of the information.  **Example**: Mask all but the last four digits of a social security number (e.g., XXX-XX-1234). |
| **Dynamic Data Masking** | **Definition**: Dynamically control and limit the exposure of confidential data in real-time during query execution. In other words, sensitive data is masked now of retrieval, just before being presented to the user (usually the masking logic is based on user roles).  **Example**: Mask salary information in query results for users without financial access rights. |
| **Randomization** | **Definition**: Introduce randomness to the values of sensitive data for creating diverse test datasets.  **Example**: Randomly adjust salary values within a specified percentage range for a group of employees. |

### **2. Data Subsetting**

Data subsetting is a technique to create a smaller yet representative subset of a production database for use in testing and development environments. There are several benefits to this technique:

1. Reduce data volume, especially in organizations with large datasets. For testing purposes, smaller data volume minimizes resource requirements and therefore reduces maintenance needs.
2. Preserve data integrity, as subsetting a dataset does not change the relationship between rows, columns, and any entities within it
3. Easily include/exclude data based on specific criteria relevant to the team’s testing needs, giving them a higher level of control. At the same time, this translates into improved efficiency in terms of data storage, transmission, and processing.

### **3. Synthetic Data Generation**

Synthetic data generation is the process of creating artificial datasets that simulate real-world data without containing any sensitive or confidential information. This approach is usually reserved only for when obtaining real data is challenging (i.e. financial, medical, legal data) or risky data (i.e. employee personal information).

In such cases, generating entirely new sets of data for testing purposes is a more practical approach. These synthetic datasets aim to simulate the original dataset as closely as possible, and that means capturing its statistical properties, patterns, and relationships.

To create new test data, you can leverage Generative AI. Simply provide the AI with clear-cut prompts for how you want your dataset to be. If you want to go above and beyond, you can custom train an AI with real-world data samples.