What is ETL Testing?

ETL is commonly associated with Data Warehousing projects but there in reality any form of bulk data movement from a source to a target can be considered ETL. Large enterprises often have a need to move application data from one source to another for data integration or data migration purposes. ETL testing is a data centric testing process to validate that the data has been tranformed and loaded into the target as expected.

Challenges in ETL Testing

ETL Testing is different from application testing because it requires a data centric testing approach. Some of the challenges in ETL Testing are:   
   - ETL Testing involves comparing of large volumes of data typically millions of records.  
   - The data that needs to be tested is in heterogeneous data sources (eg. databases, flat files).  
   - Data is often transformed which might require complex SQL queries for comparing the data.  
   - ETL testing is very much dependent on the availability of test data with different test scenarios.  
  
Although there are slight variations in the type of tests that need to be executed for each project, below are the most common types of tests that need to be done for ETL Testing.

## ETL Testing Categories

#### **METADATA TESTING**

###### **The purpose of Metadata Testing is to verify that the table definitions conform to the data model and application design specifications.**

###### **Data Type Check**

Verify that the table and column data type definitions are as per the data model design specifications.

Example: Data Model column data type is NUMBER but the database column data type is STRING (or VARCHAR).

###### **Data Length Check**

Verify that the length of database columns are as per the data model design specifications.

Example: Data Model specification for the 'first\_name' column is of length 100 but the corresponding database table column is only 80 characters long.

###### **Index/Constraint Check**

Verify that proper constraints and indexes are defined on the database tables as per the design specifications.  
   Verify that the columns that cannot be null have the 'NOT NULL' constraint.   
   Verify that the unique key and foreign key columns are indexed as per the requirement.   
   Verify that the table was named according to the table naming convention. 

Example 1: A column was defined as 'NOT NULL' but it can be optional as per the design.  
Example 2: Foreign key constraints were not defined on the database table resulting in orphan records in the child table.

###### **Metadata Naming Standards Check**

Verify that the names of the database metadata such as tables, columns, indexes are as per the naming standards.

Example: The naming standard for Fact tables is to end with an '\_F' but some of the fact tables names end with '\_FACT'.

###### **Metadata Check Across Environments**

Compare table and column metadata across environments to ensure that changes have been migrated appropriately.

Example: A new column added to the SALES fact table was not migrated from the Development to the Test environment resulting in ETL failures.

**Automate metadata testing with ETL Validator**

#### **METADATA TESTING**

###### **The purpose of Metadata Testing is to verify that the table definitions conform to the data model and application design specifications.**

###### **Data Type Check**

Verify that the table and column data type definitions are as per the data model design specifications.

Example: Data Model column data type is NUMBER but the database column data type is STRING (or VARCHAR).

###### **Data Length Check**

Verify that the length of database columns are as per the data model design specifications.

Example: Data Model specification for the 'first\_name' column is of length 100 but the corresponding database table column is only 80 characters long.

###### **Index/Constraint Check**

Verify that proper constraints and indexes are defined on the database tables as per the design specifications.  
   Verify that the columns that cannot be null have the 'NOT NULL' constraint.   
   Verify that the unique key and foreign key columns are indexed as per the requirement.   
   Verify that the table was named according to the table naming convention. 

Example 1: A column was defined as 'NOT NULL' but it can be optional as per the design.  
Example 2: Foreign key constraints were not defined on the database table resulting in orphan records in the child table.

###### **Metadata Naming Standards Check**

Verify that the names of the database metadata such as tables, columns, indexes are as per the naming standards.

Example: The naming standard for Fact tables is to end with an '\_F' but some of the fact tables names end with '\_FACT'.

###### **Metadata Check Across Environments**

Compare table and column metadata across environments to ensure that changes have been migrated appropriately.

Example: A new column added to the SALES fact table was not migrated from the Development to the Test environment resulting in ETL failures.

**Automate metadata testing with ETL Validator**

#### **DATA COMPLETENESS TESTING**

###### **The purpose of Data Completeness tests are to verify that all the expected data is loaded in target from the source. Some of the tests that can be run are : Compare and Validate counts, aggregates (min, max, sum, avg) and actual data between the source and target.**

###### **Record Count Validation**

Compare count of records of the primary source table and target table. Check for any rejected records.

Example: A simple count of records comparison between the source and target tables.   
**Source Query**  
*SELECT count(1) src\_count FROM customer*  
**Target Query**  
*SELECT count(1) tgt\_count FROM customer\_dim*

###### **Column Data Profile Validation**

Column or attribute level data profiling is an effective tool to compare source and target data without actually comparing the entire data. It is similar to comparing the checksum of your source and target data. These tests are essential when testing large amounts of data.   
  
Some of the common data profile comparisons that can be done between the source and target are:   
   Compare **unique values** in a column between the source and target  
   Compare **max, min, avg, max length, min length** values for columns depending of the data type  
   Compare **null values** in a column between the source and target  
   For important columns, compare **data distribution (frequency)** in a column between the source and target

Example 1: Compare column counts with values (non null values) between source and target for each column based on the mapping.   
**Source Query**  
*SELECT count(row\_id), count(fst\_name), count(lst\_name), avg(revenue) FROM customer*  
**Target Query**  
*SELECT count(row\_id), count(first\_name), count(last\_name), avg(revenue) FROM customer\_dim*  
  
Example 2: Compare the number of customers by country between the source and target.  
**Source Query**  
*SELECT country, count(\*) FROM customer GROUP BY country*  
**Target Query**  
*SELECT country\_cd, count(\*) FROM customer\_dim GROUP BY country\_cd*

###### **Compare entire source and target data**

Compare data (values) between the flat file and target data effectively validating 100% of the data. In regulated industries such as finance and pharma, 100% data validation might be a compliance requirement. It is also a key requirement for data migration projects. However, performing 100% data validation is a challenge when large volumes of data is involved. This is where ETL testing tools such as ETL Validator can be used because they have an inbuilt ELV engine (Extract, Load, Validate) capabile of comparing large values of data.

Example: Write a source query that matches the data in the target table after transformation.  
**Source Query**  
*SELECT cust\_id, fst\_name, lst\_name, fst\_name||','||lst\_name, DOB FROM Customer*  
**Target Query**  
*SELECT integration\_id, first\_name, Last\_name, full\_name, date\_of\_birth FROM Customer\_dim*

#### **DATA QUALITY TESTING**

###### **The purpose of Data Quality tests is to verify the accuracy of the data. Data profiling is used to identify data quality issues and the ETL is designed to fix or handle these issue. However, source data keeps changing and new data quality issues may be discovered even after the ETL is being used in production. Automating the data quality checks in the source and target system is an important aspect of ETL execution and testing.**

###### **Duplicate Data Checks**

Look for duplicate rows with same unique key column or a unique combination of columns as per business requirement.

Example: Business requirement says that a combination of First Name, Last Name, Middle Name and Data of Birth should be unique.  
**Sample query to identify duplicates**  
*SELECT fst\_name, lst\_name, mid\_name, date\_of\_birth, count(1) FROM Customer GROUP BY fst\_name, lst\_name, mid\_name HAVING count(1)>1*

###### **Data Validation Rules**

Many database fields can contain a range of values that cannot be enumerated. However, there are reasonable constraints or rules that can be applied to detect situations where the data is clearly wrong. Instances of fields containing values violating the validation rules defined represent a quality gap that can impact ETL processing.

Example: Date of birth (DOB). This is defined as the DATE datatype and can assume any valid date. However, a DOB in the future, or more than 100 years in the past are probably invalid. Also, the date of birth of the child is should not be greater than that of their parents.

###### **Data Integrity Checks**

This measurement addresses "keyed" relationships of entities within a domain. The goal of these checks is to idenfity orphan records in the child entity with a foreign key to the parent entity.  
   1. Count of records with null foreign key values in the child table  
   2. Count of invalid foriegn key values in the child table that do not have a corresponding primary key in the parent table

Example: In a data warehouse scerario, fact tables have foriegn keys to the dimension tables. If an ETL process does a full refresh of the dimension tables while the fact table is not refreshed, the surrogate foreign keys in the fact table are not valid anymore. "Late arriving dimensions" is another scenario where a foreign key relationship mismatch might occur because the fact record gets loaded ahead of the dimension record.   
**1. Count of null or unspecified dimension keys in a Fact table:**   
*SELECT count(cust\_id) FROM sales where cust\_id is null*  
  
**2. Count of invalid foriegn key values in the Fact table:**   
*SELECT cust\_id FROM sales  
minus  
SELECT s.cust\_id FROM sales s, customers c where s.cust\_id=c.cust\_id*

#### **DATA TRANSFORMATION TESTING**

###### **Data is transformed during the ETL process so that it can be consumed by applications on the target system. Transformed data is generally important for the target systems and hence it is important to test transformations. There are two approaches for testing transformations - white box testing and blackbox testing**

###### **Transformation testing using White Box approach**

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code.   
For transformation testing, this involves reviewing the transformation logic from the mapping design document and the ETL code to come up with test cases.   
  
The steps to be followed are listed below:  
   Review the source to target mapping design document to understand the transformation design   
   Apply transformations on the data using SQL or a procedural language such as PLSQL to reflect the ETL transformation logic   
   Compare the results of the transformed test data with the data in the target table.  
  
The advantage with this approach is that the test can be rerun easily on a larger source data. The disadvantage of this approach is that the tester has to reimplement the transformation logic. 

Example: In a financial company, the interest earned on the savings account is dependent the daily balance in the account for the month.   
1. Review the requirement and design for calculating the interest.   
2. Implement the logic using your favorite programming language.   
3. Compare your output with data in the target table.

###### **Transformation testing using Black Box approach**

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. For transformation testing, this involves reviewing the transformation logic from the mapping design document setting up the test data appropriately.   
  
The steps to be followed are listed below:   
   Review the requirements document to understand the transformation requirements   
   Prepare test data in the source systems to reflect different transformation scenarios  
   Come with the transformed data values or the expected values for the test data from the previous step  
   Compare the results of the transformed test data in the target table with the expected values.  
  
The advantage with this approach is that the transformation logic does not need to be reimplemented during the testing. The disadvantage of this approach is that the tester needs to setup test data for each transformation scenario and come up with the expected values for the transformed data manually. 

Example: In a financial company, the interest earned on the savings account is dependent the daily balance in the account for the month.   
1. Review the requirement for calculating the interest.   
2. Setup test data for various scenarios of daily account balance in the source system.   
3. Compare the transformed data in the target table with the expected values for the test data.

#### **ETL REGRESSION TESTING**

###### **The goal of ETL Regression testing is to verify that the ETL is producing the same output for a given input before and after the change. Any differences need to be validated whether are expected as per the changes.**

###### **Changes to Metadata**

Track changes to table metadata in the Source and Target environments. Often changes to source and target system metadata changes are not communicated to the QA and Development teams resulting in ETL and Application failures. This check is important from a regression testing standpoint.

Example 1: The length of a comments column in the source database was increased but the ETL development team was not notified. Data started getting truncaed in production data warehouse for the comments column after this change was deployed in the source system.   
  
Example 2: One of the index in the data warehouse was dropped accidentally which resulted in performance issues in reports.

###### **Automated ETL Testing**

Automating the ETL testing is the key for regression testing of the ETL particularly more so in an agile development environment. Organizing test cases into test plans (or test suites) and executing them automatically as and when needed can reduce the time and effort needed to perform the regression testing. Automating ETL testing can also eliminate any human errors while performing manual checks.

###### **Regression testing by baselining target data**

Often testers need to regression test an existing ETL mapping with a number of transformations. It may not be practical to perform an end-to-end transformation testing in such cases given the time and resource constraints. From a pure regression testing standpoint it might be sufficient to baseline the data in the target table or flat file and compare it with the actual result in such cases.   
Here are the steps:   
    Execute the ETL before the change and make a copy of the target table   
    Execute the modified ETL that needs to be regression tested  
    Compare data in the target table with the data in the baselined table to identify differences.   
    Compare the results of the transformed test data in the target table with the expected values.

Example: In the data warehouse scenario, ETL changes are pushed on a periodic basis (eg. monthly). The tester is tasked with regression testing the ETL. By following the steps outlined above, the tester can regression test key ETLs.

#### **REFERENCE DATA TESTING**

###### **Many database fields can only contain limited set of enumerated values. Instances of fields containing values not found in the valid set represent a quality gap that can impact processing.**

###### **Verify that data conforms to reference data standards**

Data model standards dictate that the values in certain columns should adhere to a values in a domain.

Example: Values in the country\_code column should have a valid country code from a Country Code domain.   
*select distinct country\_code from address   
minus  
select country\_code from country*

###### **Compare domain values across environments**

One of the challenge in maintaining reference data is to verify that all the reference data values from the development environments has been migrated properly to the test and production environments.

Example: Compare Country Codes between development, test and production environments.

###### **Track reference data changes**

Baseline reference data and compare it with the latest reference data so that the changes can be validated.

Example: A new country code has been added and an existing country code has been marked as deleted in the development environment without the approval or notification to the data steward.

#### **INCREMENTAL ETL TESTING**

###### **ETL process is generally designed to be run in a Full mode or Incremental mode. When running in Full mode, the ETL process truncates the target tables and reloads all (or most) of the data from the source systems. Incremental ETL only loads the data that changed in the source system using some kind of change capture mechanism to identify changes. Incremental ETL is essential to reducing the ETL run times and it is often used method for updating data on a regular basis. The purpose of Incremental ETL testing is to verify that updates on the sources are getting loaded into the target system properly.**

###### **While most of the data completeness and data transformation tests are relevant for incremental ETL testing, there are a few additional tests that are relevant. To start with, setup of test data for updates and inserts is a key for testing Incremental ETL.**

###### **Duplicate Data Checks**

When a source record is updated, the incremental ETL should be able to lookup for the existing record in the target table and update it. If not this can result in duplicates in the target table.

Example: Business requirement says that a combination of First Name, Last Name, Middle Name and Data of Birth should be unique.  
**Sample query to identify duplicates**  
*SELECT fst\_name, lst\_name, mid\_name, date\_of\_birth, count(1) FROM Customer GROUP BY fst\_name, lst\_name, mid\_name HAVING count(1)>1*

###### **Compare data values**

Verify that the changed data values in the source are reflecting correctly in the target data. Typically, the records updated by an ETL process are stamped by a run ID or a date of the ETL run. This date can be used to identify the newly updated or inserted records in the target system. Alternatively, all the records that got updated in the last few days in the source and target can be compared based on the incremental ETL run frequency.

Example: Write a source query that matches the data in the target table after transformation.  
**Source Query**  
*SELECT fst\_name||','||lst\_name FROM Customer where updated\_dt>sysdate-7*  
**Target Query**  
*SELECT full\_name FROM Customer\_dim where updated\_dt>sysdate-7*

###### **Data Denormalization Checks**

Denormalization of data is quite common in a data warehouse environment. Source data is denormalized in the ETL so that the report performance can be improved. However, the denormalized values can get stale if the ETL process is not designed to update them based on changes in the source data.

Example: The Customer dimension in the data warehouse is denormalized to have the latest customer address data. However, the incremental ETL for the Customer Dim was not designed to update the latest address data when the customer updates their address because it was only designed to handle the Change Capture on the Customer source table and not the Customer\_address table. The Customer address shown in the Customer Dim was good when a Full ETL was run but as the Customer Address changes come in during the Incremental ETL, the data in the Customer Dim became stale.   
  
**Source Query**  
*SELECT cust\_id, address1, address2, city, state, country FROM Customer SELECT cust\_id, address1, address2, city, state, country,   
ROW\_NUMBER( ) OVER (PARTITION BY cust\_id ORDER BY created\_date NULLS LAST) addr\_rank  
FROM Customer*  
WHERE ROW\_NUMBER( ) OVER (PARTITION BY cust\_id ORDER BY created\_date NULLS LAST) = 1  
  
**Target Query**  
*SELECT cust\_id, address1, address2, city, state, country FROM Customer\_dim*

###### **Slowly Changing Dimension Checks**

While there are different types of slowly changing dimensions (SCD), testing of and SCD Type 2 dimension presently a unique challenge since there can be multiple records with the same natural key. Type 2 SCD is designed to create a new record whenever there is a change to a set of columns. The latest record is tagged with a flag and there are start date and end date columns to indicate the period of relevance for the record. Some of the tests specific to a Type 2 SCD are listed below:   
   1. Is a new record created everytime there is a change to the SCD key columns as expected?  
   2. Is a latest record tagged as the latest record by a flag?  
   3. Are the old records end dated appropriately?

#### **ETL INTEGRATION TESTING**

###### **Once the data is transformed and loaded into the target by the ETL process, it is consumed by another application or process in the target system. For data warehouse projects, the consuming application is a BI tool such as OBIEE, Business Objects, Cognos or SSRS. For a data migration project, data is extracted from a legacy application and loaded into a new application. In a data integration project, data is being shared between two different applications usually on a regular basis. The goal of ETL integration testing is to perform an end-to-end testing of the data in the ETL process and the consuming application.**

###### **End-to-End Data Testing**

Integration testing of the ETL process and the related applications involves the following steps:   
   Setup test data in the source system  
   Execute ETL process to load the test data into the target  
   View or process the data in the target system  
   Validate the data and application functionality that uses the data

Example: Lets consider a data warehouse scenario for Case Management analytics using OBIEE as the BI tool. An executive report shows the number of Cases by Case type in OBIEE. However, during testing when the number of cases were compared between the source, target (data warehouse) and OBIEE report, it was found that each of them showed different values. As part of this testing it is important to identify the key measures or data values that can be compared across the source, target and consuming application.

#### **ETL PERFORMANCE TESTING**

###### **Performance of the ETL process is one of the key issues in any ETL project. Often development environments do not have enough source data for performance testing of the ETL process. This could be because the project has just started and the source system only has small amount of test data or production data has PII information which cannot be loaded into the test database without scrubbing. The ETL process can behave differently with different volumes of data.**

Example 1: A lookup might perform well when the data is small but might become a bottle neck that slowed down the ETL task when there is large volume of data. What can make it worse is that the ETL task may be running by itself for hours causing the entire ETL process to run much longer than the expected SLA.   
  
Example 2: An incremental ETL task was updating more records than it should. When the data volumes were low in the target table, it performed well but when the data volumes increased, the updated slowed down the incremental ETL tremendously.

###### **End-to-End Data Testing**

Integration testing of the ETL process and the related applications involves the following steps:   
   Estimate expected data volumes in each of the source table for the ETL for the next 1-3 years.  
   Setup test data for performance testing either by generating sample data or making a copy of the production (scrubbed) data.  
   Execute Full ETL process to load the test data into the target.  
   Review each individual ETL task (workflow) run times and the order of execution of the ETL. Revisit ETL task dependencies and reorder the ETL tasks so that the tasks run in parallel as much as possible.  
   Setup test data for incremental ETL process with the data change volumes as expected during an incremental ETL.  
   Executing incremental ETL. Review ETL task load times and the order of execution of the tasks to identify bottlenecks.

### Difference between Database and Data Warehouse Testing

There is a popular misunderstanding that [database testing](http://www.softwaretestinghelp.com/category/database-testing/) and [data warehouse](http://en.wikipedia.org/wiki/Data_warehouse) is similar while the fact is that both hold different direction in testing.

* Database testing is done using a smaller scale of data normally with OLTP (Online transaction processing) type of databases while data warehouse testing is done with large volume with data involving OLAP (online analytical processing) databases.
* In database testing normally data is consistently injected from uniform sources while in data warehouse testing most of the data comes from different kind of data sources which are sequentially inconsistent.
* We generally perform the only CRUD (Create, read, update and delete) operation in database testing while in data warehouse testing we use read-only (Select) operation.
* Normalized databases are used in DB testing while demoralized DB is used in data warehouse testing.

There is a number of universal verifications that have to be carried out for any kind of data warehouse testing. Below is the list of objects that are treated as essential for validation in ETL testing:  
– Verify that data transformation from source to destination works as expected  
– Verify that expected data is added in target system  
– Verify that all DB fields and field data is loaded without any truncation  
– Verify data checksum for record count match  
– Verify that for rejected data proper error logs are generated with all details  
– Verify NULL value fields  
– Verify that duplicate data is not loaded  
– Verify data integrity

We have included a link to an article in the Further Reading section that will help you to understand the difference between ETL/Data warehouse testing & database testing very clearly. The article is named as “ ETL vs. DB Testing – A Closer Look at ETL Testing Need, Planning and ETL Tools”

### ETL Testing Challenges:

ETL testing is quite different from conventional testing. There are many challenges we faced while performing data warehouse testing. Here is the list of few ETL [testing challenges](http://www.softwaretestinghelp.com/manual-and-automation-testing-challenges/) I experienced on my project:

– Incompatible and duplicate data.  
– Loss of data during ETL process.  
– Unavailability of the inclusive test bed.  
– Testers have no privileges to execute ETL jobs by their own.  
– Volume and complexity of data are very huge.  
– Fault in business process and procedures.  
– Trouble acquiring and building test data.  
– Unstable testing environment.  
– Missing business flow information.

Data is important for businesses to make the critical business decisions. ETL testing plays a significant role validating and ensuring that the business information is exact, consistent and reliable. Also, it minimizes the hazard of data loss in production.

Hope these tips will help ensure your ETL process is accurate and the data warehouse build by this is a competitive advantage for your business.

**Further Reading:**

* [How to Perform ETL Testing Using Informatica PowerCenter Tool](http://www.softwaretestinghelp.com/etl-testing-usin%E2%80%A6powercenter-tool/)
* [ETL vs. DB Testing – A Closer Look at ETL Testing Need, Planning and ETL Tools](http://www.softwaretestinghelp.com/etl-testing-vs-db-testing/)
* [The 4 Steps to Business Intelligence (BI) Testing: How to Test Business Data](http://www.softwaretestinghelp.com/business-intelligence-bi-testing/)
* [Top 4 ETL Testing Tools](http://www.softwaretestinghelp.com/top-4-etl-testing-tools/)

### Difference between Database and Data Warehouse Testing

There is a popular misunderstanding that [database testing](http://www.softwaretestinghelp.com/category/database-testing/) and [data warehouse](http://en.wikipedia.org/wiki/Data_warehouse) is similar while the fact is that both hold different direction in testing.

* Database testing is done using a smaller scale of data normally with OLTP (Online transaction processing) type of databases while data warehouse testing is done with large volume with data involving OLAP (online analytical processing) databases.
* In database testing normally data is consistently injected from uniform sources while in data warehouse testing most of the data comes from different kind of data sources which are sequentially inconsistent.
* We generally perform the only CRUD (Create, read, update and delete) operation in database testing while in data warehouse testing we use read-only (Select) operation.
* Normalized databases are used in DB testing while demoralized DB is used in data warehouse testing.

There is a number of universal verifications that have to be carried out for any kind of data warehouse testing. Below is the list of objects that are treated as essential for validation in ETL testing:  
– Verify that data transformation from source to destination works as expected  
– Verify that expected data is added in target system  
– Verify that all DB fields and field data is loaded without any truncation  
– Verify data checksum for record count match  
– Verify that for rejected data proper error logs are generated with all details  
– Verify NULL value fields  
– Verify that duplicate data is not loaded  
– Verify data integrity

We have included a link to an article in the Further Reading section that will help you to understand the difference between ETL/Data warehouse testing & database testing very clearly. The article is named as “ ETL vs. DB Testing – A Closer Look at ETL Testing Need, Planning and ETL Tools”

### ETL Testing Challenges:

ETL testing is quite different from conventional testing. There are many challenges we faced while performing data warehouse testing. Here is the list of few ETL [testing challenges](http://www.softwaretestinghelp.com/manual-and-automation-testing-challenges/) I experienced on my project:

– Incompatible and duplicate data.  
– Loss of data during ETL process.  
– Unavailability of the inclusive test bed.  
– Testers have no privileges to execute ETL jobs by their own.  
– Volume and complexity of data are very huge.  
– Fault in business process and procedures.  
– Trouble acquiring and building test data.  
– Unstable testing environment.  
– Missing business flow information.

Data is important for businesses to make the critical business decisions. ETL testing plays a significant role validating and ensuring that the business information is exact, consistent and reliable. Also, it minimizes the hazard of data loss in production.

Hope these tips will help ensure your ETL process is accurate and the data warehouse build by this is a competitive advantage for your business.

**Further Reading:**

* [How to Perform ETL Testing Using Informatica PowerCenter Tool](http://www.softwaretestinghelp.com/etl-testing-usin%E2%80%A6powercenter-tool/)
* [ETL vs. DB Testing – A Closer Look at ETL Testing Need, Planning and ETL Tools](http://www.softwaretestinghelp.com/etl-testing-vs-db-testing/)
* [The 4 Steps to Business Intelligence (BI) Testing: How to Test Business Data](http://www.softwaretestinghelp.com/business-intelligence-bi-testing/)
* [Top 4 ETL Testing Tools](http://www.softwaretestinghelp.com/top-4-etl-testing-tools/)

.