**Difference between ETL and ELT?**

* ETL means first we extract a Data then we transform a data and eventually Load a data
* ELT means first we extract a data then load it into staging area and then transform a data but with SQL Expression tasks not with transformations task in your fact and dimension tables.
* ELT is better as DB engine can perform transformations more better then Informatica packages.
* So in short, when using ETL the transformations are processed by the ETL tools, while in ELT the transformations are processed by the target datasource.
* Pushdown is the way to implement ELT

**Crontab Syntax:**

\*     \*   \*  \*   \*  command to be executed

-     -    -   -  -

|     |     |   |    |

|     |     |   |    +----- day of week (0 - 6) (Sunday=0)

|     |     |   +------- month (1 - 12)

|     |     +--------- day of month (1 - 31)

|     +----------- hour (0 - 23)

+------------- min (0 - 59)

**What is MDM? How to implement?**

MDM is unified data model (Master data Management).Instead of having varied snapshot of important information it is advisable to have Master data for that entity and refer to the same

- Identify Master data entity with below points

- Data needs to be less volatile compare to transaction data (Which changes less compare to facts)

- Cardinality is more (Number of records are more for i.e. customers)

- Data changes over time then it should be qualified for MDM

- Reuse of Data from MDM (Reusability is high)

**Ralph vs Inmon**

* Kimball uses the dimensional model such as star schemas or snowflakes to organize the data in dimensional data warehouse while Inmon uses ER model in enterprise data warehouse.
* Inmon only uses dimensional model for data marts only while Kimball uses it for all data
* Inmon uses data marts as physical separation from enterprise data warehouse and they are built for departmental uses. While in Kimball’s architecture, it is unnecessary to separate the data marts from the dimensional data warehouse.

In dimensional data warehouse of Kimball, analytic systems can access data directly. While in Inmon’ s architecture, analytic systems can only access data in enterprise data warehouse via data marts.

**Steps to design DW from OLTP**

In general, building any data warehouse consists of the following steps:

**Flow of Data:**

* Landing(Extraction from Source)
* Staging(Data sits in one format from source):Data staging is an area where you hold the data temporary on data warehouse server. Data staging includes following steps
* Source data extraction and data transformation ( restructuring )
* Data transformation (data cleansing, value transformation )

**Extracting** the transactional data from the data sources into a staging area

* Transforming the transactional data
* Loading the transformed data into a dimensional database
* Building pre-calculated summary values to speed up report generation
* Building (or purchasing) a front-end reporting tool

**Analysis**:

* Requirement gathering for BI reports and business models.
* Understanding the objective for which we are designing DW.

**Design**:

* Identify the source databases or data, understanding and defining hierarchies.
* Identifying measures, identifying dimensions and their hierarchies, designing and creating physical models by defining schema of fact and dimension tables.
* Design Staging area for Data.
* Granularity level of dataware house - Requirement speccifci
* CDC of given data.

**Data presentation**

* Area where data will be in normalized form.

**Data Gathering**:

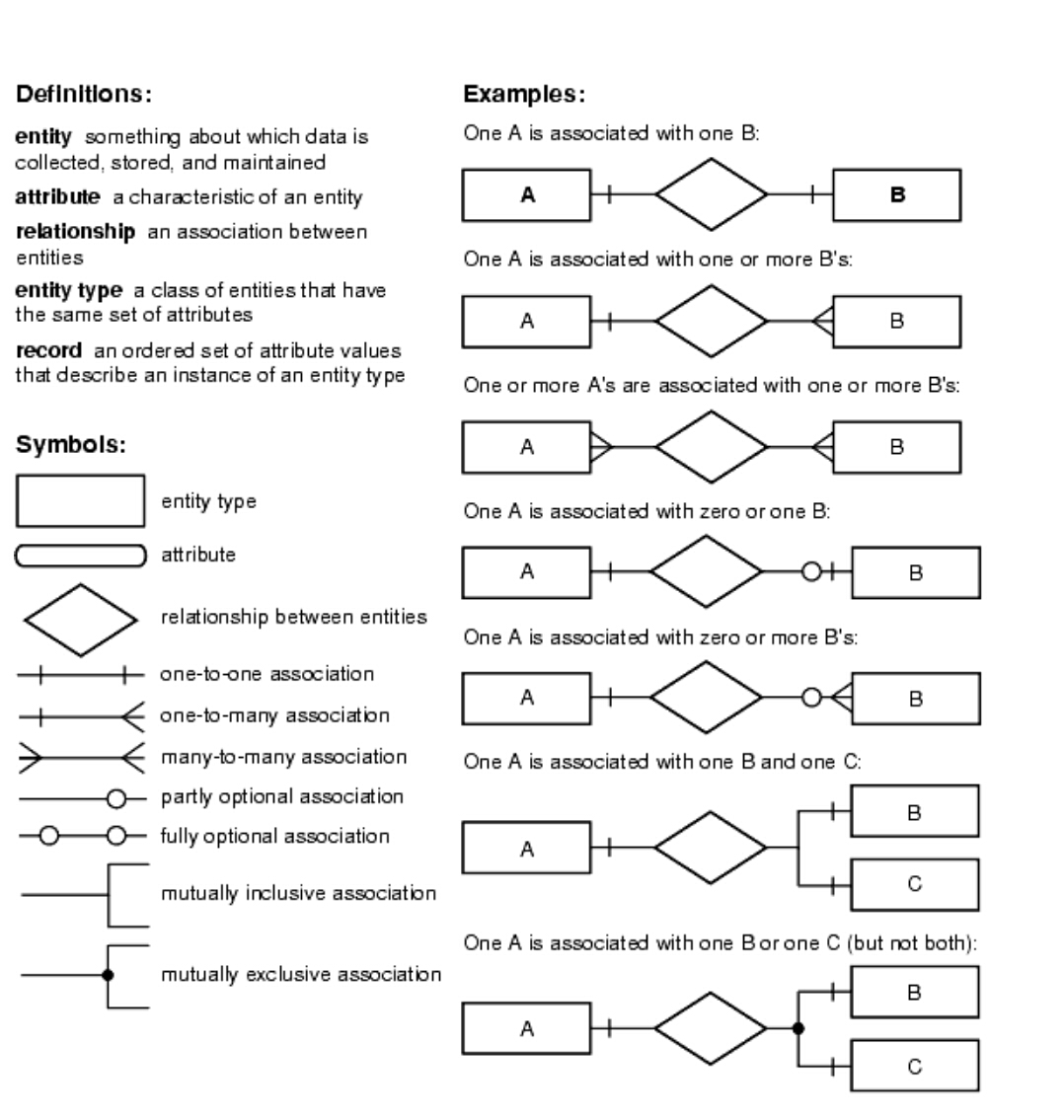
* This is one of the important step in designing the good Data Warehouse system.This is step is nothing but ETL development i.e., Extraction Transformation and Loading. Here the Data is Extracted and by using the transformations we convert this data to normalized form.The normalized format is not so good for understanding purpose.So we go for Data Presentation Area. Data gathering phase involves extracting and cleaning data to prepare it for loading into the warehouse. This phase is called low level design.

**Data mart creation**:

* Here we convert the Normalized model to Dimensional model where the user can understand very easily. Here the data will be in data marts. By combining the data marts we can have a Data Warehouse.
* **Conformed Dimension**: In data warehousing, a conformed dimension is a dimension that has the same meaning to every fact with which it relates and which can be connected to multiple DW and give the same meaning.
* **Conformed Fact**: A conformed fact is a shared fact that is designed to be used in the same way. Across multiple data marts.
* **Junk Dimension:** Indicator fields like Yes/No values which wont be a good choice for dimension table can be grouped and maintained in one another table which can be term as a Junk dimension.
* **Mini Dimension:** A rapidly changing dimension can grow very large, perhaps too large.We can handle that by creating mini dimension
* **Degenerate Dimension:** Dimension key which does not have its own dimension can be termed as Degenerate dimension. i.e. Invoice No
* **Rapidly Changing Dimension:** Frequently changing fields can be moved to separate table

**SCD Type:**

* **Type 1:** overwrite
* **Type 2**: add new row (With Is Active or Effective Dates Column)
* **Type 3**: add new attribute (Add new column to capture the change i.e having previous stateof customer and current state of customer)
* **Type 4**: add history table

**ER Diagram points:**

**Benefits of Surrogate Keys:**

* Performance: Surrogate key is always numeric key unlike any production key which is alphanumeric so it can be good qualifier to create index and use in join statements
* No Possibilities of duplication and maintenance is quite easy as no composite key creation required.
* Immutability: Keys can’t be changed
* Stay as it is with requirement changes( Natural keys can become duplicate on merger surrogate cant)
* Surrogate key assignments
  + ODS (Transformations)
  + DW(Data presentation area)

**Fact less Fact**: Facts which does not contain any measures and Count keyword only we can operate.This table only contains IDs which links to dimension.

**Additive Facts:**

**Non-Additive facts:**

**Semi-Additive facts:**

Informatica:

**Informatica Concepts to look:**

-error handling

-Identifying bottlenecks

-Source and Destination

-Nested join and Hash Joins

-Multiple sources in SSIS jobs

-How to optimize job which take huge amount of time?

-Challenging job

-Incremental load

-Protection level

-Deployment

**Assignments of Informatica:**

1. ~~Load a flat file to Database~~
2. ~~Load a database table to Flat file – Done~~
3. ~~Load a database table to Multiple targets - Done~~
4. Avoid double loads when jobs are run twice.
5. ~~Add calculated column with functions and expressions and load into Target - Done~~
6. Handle erroneous data into other table and clean into target table.
7. Join data from various sources through transformation and load into single target
8. Incremental Load of Data from Flat file
9. Change data capture implementation
10. Target Destructive loads
11. Update Existing Record or if not exists then insert new.
12. Remove Duplicate records from FF apart from using SORTER transformation.
13. Create query in source qualifier and create job
14. Filter Transformation
15. Router Transformation
16. Differences between Filter and Router
17. Sequence Generator Transformation
18. Cache
19. Sorter Transformation
20. Aggregator Transformation
21. Implement SCD Job Type 2
22. Implement MDM Job
23. Multigroup Transformations
24. Normalizer Transformation
25. Pivot and unpivot data
26. Compare previous value in expression transformation
27. Geenrate running total

**Components in informatica:**

* **Workflow:** A workflow is a set of instruction on how to execute tasks such as sessions (mappings), sending email, os commands (shell/batch) etc.
* **Session:** Create a session for each mapping which performs ETL operation.
* **Worklet:** A worklet is a set of instructions that can be reused in several workflows.
* **Mapping:** Set of transformations to execute ETL jobs.

**Mapping and Mapplets**

**Workflow and worklets**

**Sessions**

**Difference between mapplet and mappings:**

* **Mapplets** are re-usable components and is a set of transformation / Business logic which can be reused across various mappings. **Mapping** is a also a set of rules / logic defined and is a re-usable component. But we cannot call a mapping within another mapping.
* Also a mapplet can contain sources but cannot contain a Target, where a mapping is expected to have a target.
* Mapping is complete object and mapplet you need a mapping to execute the flow as mapplet doesn’t have target to load the data.

**Note:**

**Session properties always override the mapping properties**

**Transformations in Informatica:**

| **Transformation** | **Type** | **Description** |
| --- | --- | --- |
| [Aggregator](http://www.howtointegratedata.com/aggregator-transformation-in-informatica/) | Active | Performs aggregate calculations.   * Please note that the session will fail if it is configured to use sorted input but input data is not sorted accurately. * **What will Informatica Aggregator Transformation return if you do not configure Group By port?**   Informatica will return last input row read from the incoming data in the pipeline.   * **Explain two Informatica Aggregator Transformation caches?** Here are two caches created and maintained by an Aggregator transformation – Data Cache and Index Cache. Index Cache preserves the Group By port values and Data Cache preserves the rest such as aggregate calculations, variable and other input values. * **Verify the incremental aggregation settings in the session properties.** * **How to Configure incremental aggregation?**   You have to use aggregator transformation. You can find the incremental aggregation in the session properties tab under the performance section.  **What is incremental aggregation?**  To Store aggregated values at target without doing complete load of data. Partial aggregation is stored in Data and Index file  **Notes:**   * Single Level Aggregate Function: MAX(SAL) * Nested Aggregate Function: MAX( COUNT( ITEM )) * Nested Aggregate Functions * In Aggregator transformation, there can be multiple single level functions or multiple nested functions.An Aggregator transformation cannot have both types of functions together.   MAX( COUNT( ITEM )) is correct.MIN(MAX( COUNT( ITEM ))) is not correct. It can also include one aggregate function nested within another aggregate function   * When grouping data, the Aggregator transformation outputs the last row of each group unless otherwise specified. |
| Application Source Qualifier | Active | Represents the rows that the Integration Service reads from an application, such as an ERP source, when it runs a session. |
| Custom | Active or Passive | Calls a procedure in a shared library or DLL. |
| Data Masking | Passive | Replaces sensitive production data with realistic test data for non-production environments.  1) Key Masking  Masks the key value and produces deterministic results for a combination of input key value, masking rule and seed value.  2) Substitution Masking  Replaces the input column data with a value from a dictionary file or table  3) Dependent Masking  Replaces the value of the input column based on the value of another column  4) Random Masking  Replaces input column value with random values for same source data and masking rule  5) Expression Masking  Uses informatica transformation functions to mask and replace data in source columns  6) Special Mask formats  Replaces realistic fake values for fields like credit cards, addresses , and social security numbers etc.  7) No Masking  Default is No masking |
| [Expression](http://www.howtointegratedata.com/expression-transformation-in-informatica/) | Passive | Calculates a value. |
| External Procedure | Passive | Calls a procedure in a shared library or in the COM layer of Windows. |
| [Filter](http://www.howtointegratedata.com/filter-transformation-in-informatica/) | Active | Filters data. |
| HTTP | Passive | Connects to an HTTP server to read or update data. |
| Input | Passive | Defines mapplet input rows. Available in the Mapplet Designer. |
| [Joiner](http://www.howtointegratedata.com/joiner-transformation-informatica/) | Active | Joins data from different databases or flat file systems.   * Only 2 Data source can be joins * Can join heterogeneous sources * Master outer join –Right ( All Detail Records + Master recrods) * Detail Outer Join – left (All master records + detail records) * Full outer – Matched and not matched both * Normal – Inner   In Joiner, one table is called as MASTER and other as DETAIL.   * MASTER table is always cached. We can make any table as MASTER. * Edit Joiner Transformation -> Ports Tab -> Select M for Master table.   Table with less number of rows should be made MASTER to improve Performance.  **Notes:**   * You cannot use joiner transformation when the input pipeline contains an update strategy transformation. * You cannot connect a sequence generator transformation directly to the joiner transformation. * The Joiner transformation does not match null values. * Then the Char value is “abcd” padded with 36 blank spaces, and the Power Center Server does not join the two fields because the Char field contains trailing spaces. |
| Lookup | Active or Passive | Look up and return data from a flat file, relational table, view, or synonym.  **Active** : As we can select multiple values during join so more rows  **Passive**: If value found then return value or Null  **Unconnected:** If not connected in flow  **Connected:** connected in flow  **Static Cache:** Stationary cache which is created once and we cannot insert and update data on this.  **Dynamic Cache:** In Dynamic Cache we can insert or update rows in the cache when we pass the rows. The dynamic cache is synchronized with the target  **Persistence Cache:** It saves persistent cache on system and reuse next time. It is created as named look up cache.  **Shared Cache:** Cache is used for multiple transformations and mapping and Informatica release and allocates the memory as and when cache is used. Unnamed cache is shared between transformations in the same mapping and named cache between transformations in the same or different mappings.  **Recache from database:**  If the persistent cache is not synchronized with the lookup table you can configure the lookup transformation to rebuild the lookup cache.  **Notes:**   * Suppressing default ORDER BY clause in Lookup or modifying it. * If the cache file size of your lookup table is more than 2GB, most likely Informatica will create multiple cache files for one lookup wherein maximum file size for each file will be 2GB.If those are data cache files, Informatica will name them as .dat1, .dat2, .dat3 … etc. and corresponding index cache files will be named as .idx1, .idx2, .idx3 … etc. My personal opinion is breaking one single data or index cache file into multiple files may slow down the lookup performance. Hence if your lookup cache size is more than 2GB then if possible consider the option of joining the lookup source table in the database level itself instead of building lookup cache. * Unconnected Lookup can be used with condition to avoid look up overheads * Informatica lookup transformation by default evaluates Null = Null condition to be true. This means if you do not explicitly take care of passing Not Null values to lookup input ports then the lookup might return random records from the lookup source. * Dynamic look up can be done on Relational Source only |
| Normalizer | Active | Source qualifier for COBOL sources. Can also use in the pipeline to normalize data from relational or flat file sources.   * Active and Connected Transformation. * We can also use the Normalizer transformation with relational sources to create **multiple rows from a single row** of data. * IF you want to convert multiple rows into single row then use aggregator. * Source qualifier for COBOL sources * Generate single record multiple times. * Convert Rows to Column (Pivoting) * GKID and GCID:   + GKID: generate sequence number each time processes the source row.   + GCID: Generated column number- The GCID is an index for the instance of the multiple-occurring data. |
| Output | Passive | Defines mapplet output rows. Available in the Mapplet Designer. |
| [Rank](http://www.howtointegratedata.com/rank-transformation-in-informatica/) | Active | Limits records to a top or bottom range. |
| [Router](http://www.howtointegratedata.com/router-transformation-in-informatica/) | Active | Routes data into multiple transformations based on group conditions. |
| [Sequence Generator](http://www.howtointegratedata.com/sequence-generator-transformation-in-informatica/) | Passive | Generates primary keys. |
| [Sorter](http://www.howtointegratedata.com/sorter-transformation-in-informatica/) | Active | Sorts data based on a sort key.   * Sorter transformation can’t be used for more than one source. * It is an active transformation because it removes the duplicates from the key and consequently changes the number of rows. |
| [Source Qualifier](http://www.howtointegratedata.com/source-qualifier-transformation-in-informatica/) | Active | Represents the rows that the Integration Service reads from a relational or flat file source when it runs a session. |
| SQL | Active or Passive | Executes SQL queries against a database.   * Can Process insert, update, delete and select. * Dynamically takes DB connection * It can DDL,DML,TCL,GRANT and SELECT   **Script Mode:**   * Passive Transformation * It has 3 ports enable by default: ScriptName, ScriptResult and ScriptError. * The SQL transformation in script mode does not output more than one row of data for each input row.   **Query Mode:**   * Passive Transformation * It has 3 ports enable by default: ID,SET\_STATMENT and SQLError. * Dynamic Connection property gives you port Logical connection which can be custom |
| Stored Procedure | Passive | Calls a stored procedure.  Can be connected and unconnected.  Use of Stored Procedure in mapping:   * Check the status of a target database before loading data into it. * Determine if enough space exists in a database. * Perform a specialized calculation. * Drop and recreate indexes. Mostly used for this in projects.   We can configure it at below options also:   * Pre-load/post-load of the Source * Pre-load/post-load of the Target   Above options will run procedure when session calls source or target from the mapping |
| Transaction Control | Active | Defines commit and rollback transactions.  Transaction Control Transformation is used in a mapping to control transactions according to the user-defined conditions.  Use expression transformation to form the condition.  Built-in variables in the Expression Editor when we create a transaction control expression:    **TC\_CONTINUE\_TRANSACTION**. The PowerCenter Server does not perform any transaction change for this row. This is the default value of the expression.  **TC\_COMMIT\_BEFORE**. The PowerCenter Server commits the transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.  **TC\_COMMIT\_AFTER**. The PowerCenter Server writes the current row to the target, commits the transaction, and begins a new transaction. The current row is in the committed transaction.  **TC\_ROLLBACK\_BEFORE**. The PowerCenter Server rolls back the current transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.  **TC\_ROLLBACK\_AFTER**. The PowerCenter Server writes the current row to the target, rolls back the transaction, and begins a new transaction. The current row is in the rolled back transaction.  Used for:   * Generating dynamic flat files on output |
| [Union](http://www.howtointegratedata.com/informatica-union-transformation/) | Active | Merges data from different databases or flat file systems.   * Informatica Union is always union all transformation * Union transformation should have a similar source. * Input and output source should be same. * Use filter with union all transformation for unique records * Works with multiple sources * It can take input from more than 2 transformations |
| Unstructured Data | Active or Passive | Transforms data in unstructured and semi-structured formats. |
| [Update Strategy](http://www.howtointegratedata.com/update-strategy-transformation-in-informatica/) | Active | Determines whether to insert, delete, update, or reject rows.   * Should have RDBMS as target   In Power Center, we set the update strategy at two different levels:   * Within a session * Within a Mapping * if you want to perform TARGET updates   In such cases the UPDATE\_OVERRIDE property of TARGET Object can be useful.  UPDATE EMPL\_POST\_HIST  SET POST = :TU.POST  , UPDATE\_DATE = :TU.UPDATE\_DATE  WHERE EMPL = :TU.EMPL |
| XML Generator | Active | Reads data from one or more input ports and outputs XML through a single output port. |
| XML Parser | Active | Reads XML from one input port and outputs data to one or more output ports. |
| XML Source Qualifier | Active | Represents the rows that the Integration Service reads from an XML source when it runs a session. |

**How to remove duplicate data from flat file informatica?**

* Use Sorter Transformation and distinct property
* If RDBMS use Source Qualifier distinct property
* Use aggregator with all column as keys
* Expression transformation and variable to mark duplicates and fitler to remove it

**Difference between Joiner and Look up?**

* In lookup we can override the query but in joiner we cannot.
* In lookup we can provide different types of operators like – “>,<,>=,<=,!=” but in joiner only “= “ (equal to )operator is available.
* In lookup we can restrict the number of rows while reading the relational table using lookup override but in joiner we cannot restrict the number of rows while reading.
* In joiner we can join the tables based on- Normal Join, Master Outer, Detail Outer and Full Outer Join but in lookup this facility is not available .Lookup behaves like Left Outer Join of database.

**How to create output file dynamically at job run time?**

* Use transaction control transformation and previous record check and select commit option and target as flat file which will generate file for each value.

**How to run Informatica Workflow using Unix command**

To run the workflow first need to got the command line and go to the folder where the executable is installed. Mostly you can find the pmcmd command in \server\bin\directory.

Syntax:

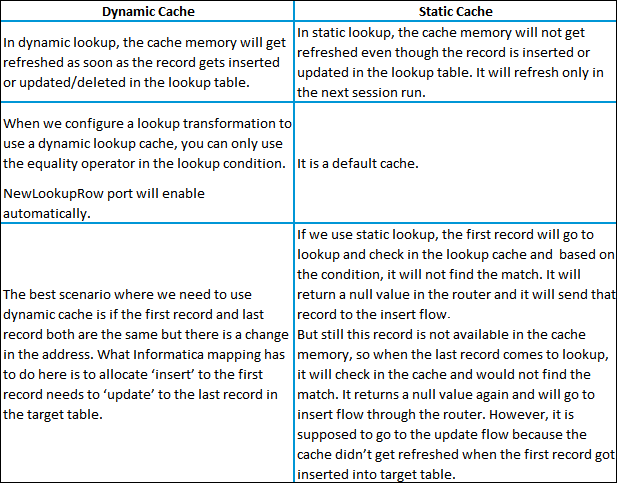
pmcmd startworkflow -sv <Integration Service Name> -d <Domain Name> -u <Integration Service Username> -p <Password> -f <Folder Name> <Workflow>

pmcmd startworkflow -sv qpd\_Service -d qpd\_Domain -u Administrator -p admin -f Alex wf\_sales\_tax

**What are the differences between Connected and Unconnected Lookup?**

* Connected lookup participates in the mapping(dataflow), just like any other transformation. Unconnected lookup is used when a lookup function is used instead in an expression transformation in the mapping in which case the lookup does not appear in the main flow (dataflow) of the mapping.
* Connected lookup can return more than one value(output port) whereas an Unconnected lookup gives only one output port. Unconnected lookups are reusable.
* We can delete lookup ports from a relational lookup if the mapping does not use the lookup ports which will give us performance gain. But if the lookup source is a flat file then deleting of lookup ports fails the session.

[**https://dwbi.org/etl/informatica/138-dynamic-lookup-cache**](https://dwbi.org/etl/informatica/138-dynamic-lookup-cache)



**Dynamic Look up Cache:**

* You cannot share the cache between a dynamic Lookup transformation and static Lookup transformation in the same target load order group.
* You can create a dynamic lookup cache from a relational table, flat file, or source qualifier transformation.
* The Lookup transformation must be a connected transformation.
* Use a persistent or a non-persistent cache.
* If the dynamic cache is not persistent, the Integration Service always rebuilds the cache from the database, even if you do not enable Re-cache from Lookup Source.
* When you synchronize dynamic cache files with a lookup source table, the Lookup transformation inserts rows into the lookup source table and the dynamic lookup cache. If the source row is an update row, the Lookup transformation updates the dynamic lookup cache only.
* You can only create an equality lookup condition. You cannot look up a range of data in dynamic cache.
* Associate each lookup port that is not in the lookup condition with an input port, sequence ID, or expression.
* Use a Router transformation to pass rows to the cached target when the NewLookupRow value equals one or two. Use the Router transformation to drop rows when the NewLookupRow value equals zero, or you can output those rows to a different target.
* Verify that you output the same values to the target that the Integration Service writes to the lookup cache. When you choose to output new values on update, only connect lookup/output ports to the target table instead of input/output ports. When you choose to output old values on update, add an Expression transformation after the Lookup transformation and before the Router transformation. Add output ports in the Expression transformation for each port in the target table and create expressions to ensure you do not output null input values to the target.
* When you use a lookup SQL override, map the correct columns to the appropriate targets for lookup.
* When you add a WHERE clause to the lookup SQL override, use a Filter transformation before the Lookup transformation. This ensures the Integration Service inserts rows in the dynamic cache and target table that match the WHERE clause.
* When you configure a reusable Lookup transformation to use a dynamic cache, you cannot edit the condition or disable the Dynamic Lookup Cache property in a mapping.
* Use Update Strategy transformations after the Lookup transformation to flag the rows for insert or update for the target.
* Use an Update Strategy transformation before the Lookup transformation to define some or all rows as update if you want to use the Update Else Insert property in the Lookup transformation.
* Set the row type to Data Driven in the session properties.
* Select Insert and Update as Update for the target table options in the session properties.

**How many types of transformations supported by sorted input?**

Aggregator Transformation (Not mandatory)

Joiner Transformation (Not mandatory)

Lookup Transformation support sorted input so that it will increase the session performance.

**Name 4 output files that Informatica server creates during session running:**

* **Informatica server log**: Informatica server (on UNIX) creates a log for all status and error messages (default name: pm.server.log). It also creates an error log for error messages. These files will be created in Informatica home directory.
* **Session log file**: Informatica server creates session log file for each session. It writes information about session into log files such as initialization process, creation of sql commands for reader and writer threads, errors encountered and load summary. The amount of detail in session log file depends on the tracing level that you set.
* **Session detail file**: This file contains load statistics for each target in mapping. Session detail includes information such as table name, number of rows written or rejected. You can view this file by double clicking on the session in monitor window.
* **Performance detail file**: This file contains information known as session performance details which helps you where performance can be improved. To generate this file select the performance detail option in the session property sheet.
* **Reject file:** This file contains the rows of data that the writer does not write to targets.
* **Control file**: Informatica server creates control file and a target file when you run a session that uses the external loader. The control file contains the information about the target flat file such as data format and loading instructions for the external loader.
* **Post session email**: Post session email allows you to automatically communicate information about a session run to designated recipients. You can create two different messages. One if the session completed successfully the other if the session fails.
* **Indicator file**: If u use the flat file as a target, you can configure the informatica server to create indicator file. For each target row, the indicator file contains a number to indicate whether the row was marked for insert,update,delete or reject.
* **Output file**: If session writes to a target file, the informatica server creates the target file based on file properties entered in the session property sheet.
* **Cache files:** When the informatica server creates memory cache it also creates cache files. For the following circumstances informatica server creates index and datacache files.
  + Aggregator transformation
  + Joiner transformation
  + Rank transformation
  + Lookup transformation

**What is the difference between stop and abort?**

Stop command immediately kills the reading process and doesn’t have any timeout period.

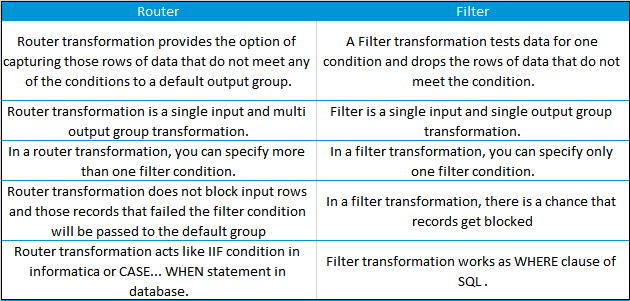
Abort command gives a time out period of 60 secs to the Informatica server to finish the dtm process else it kills the dtm process.

**G. What is Update Override? What are the differences between SQL Override and Update Override?**

**Update Override** – It is an option available in TARGET instance. By default, Target table is updated based on Primary key values. To update the Target table on non-primary key values, we can generate the default Query and override the Query according to the requirement. Suppose for example, if we want to update the record in target table, when a column value=’AAA’ then, we can include this condition in Where clause of default Query.

**Coming to SQL override** – It is an option available in Source Qualifier and Lookup transformation where we can include joins, filters, Group by and Order by.

**Filter and Router**



**What is the difference between active and passive transformation?**

**Active Transformation**:- An active transformation can perform any of the following actions:

Change the number of rows that pass through the transformation: For instance, the Filter transformation is active because it removes rows that do not meet the filter condition.

Change the transaction boundary: For e.g., the Transaction Control transformation is active because it defines a commit or roll back transaction based on an expression evaluated for each row.

Change the row type: For e.g., the Update Strategy transformation is active because it flags rows for insert, delete, update, or reject.

**Passive Transformation**: A passive transformation is one which will satisfy all these conditions:

Does not change the number of rows that pass through the transformation

Maintains the transaction boundary

Maintains the row type

**When do you use SQL override in a lookup transformation?**

You should override the lookup query in the following circumstances:

* Override the ORDER BY clause. Create the ORDER BY clause with fewer columns to increase performance. When you override the ORDER BY clause, you must suppress the generated ORDER BY clause with a comment notation.  
  Note: If you use pushdown optimization, you cannot override the ORDER BY clause or suppress the generated ORDER BY clause with a comment notation.
* A lookup table name or column names contains a reserved word. If the table name or any column name in the lookup query contains a reserved word, you must ensure that they are enclosed in quotes.
* Use parameters and variables. Use parameters and variables when you enter a lookup SQL override. Use any parameter or variable type that you can define in the parameter file. You can enter a parameter or variable within the SQL statement, or use a parameter or variable as the SQL query. For example, you can use a session parameter, $ParamMyLkpOverride, as the lookup SQL query, and set $ParamMyLkpOverride to the SQL statement in a parameter file. The designer cannot expand parameters and variables in the query override and does not validate it when you use a parameter or variable. The integration service expands the parameters and variables when you run the session.
* A lookup column name contains a slash (/) character. When generating the default lookup query, the designer and integration service replace any slash character (/) in the lookup column name with an underscore character. To query lookup column names containing the slash character, override the default lookup query, replace the underscore characters with the slash character, and enclose the column name in double quotes.
* Add a WHERE clause. Use a lookup SQL override to add a WHERE clause to the default SQL statement. You might want to use the WHERE clause to reduce the number of rows included in the cache. When you add a WHERE clause to a Lookup transformation using a dynamic cache, use a Filter transformation before the Lookup transformation to pass rows into the dynamic cache that match the WHERE clause.  
  Note: The session fails if you include large object ports in a WHERE clause.
* Other. Use a lookup SQL override if you want to query lookup data from multiple lookups or if you want to modify the data queried from the lookup table before the Integration Service caches the lookup rows. For example, use TO\_CHAR to convert dates to strings.

**What are data driven sessions?**

When you configure a session using update strategy, the session property data driven instructs Informatica server to use the instructions coded in mapping to flag the rows for insert, update, delete or reject. This is done by mentioning DD\_UPDATE or DD\_INSERT or DD\_DELETE in the update strategy transformation.

**What is the use of source qualifier?**

The source qualifier transformation is an active, connected transformation used to represent the rows that the integrations service reads when it runs a session. **The source qualifier transformation converts the source data types to the Informatica native data types**. So, you should not alter the data types of the ports in the source qualifier transformation.

**What are the different ways to filter rows using Informatica transformations?**

* Source Qualifier
* Joiner
* Filter
* Router

**What are the different transformations where you can use a SQL override?**

* Source Qualifier
* Lookup
* Target

**State the differences between SQL Override and Lookup Override?**

* Lookup Override is used to limit the number of lookup rows to avoid the whole table scan by saving the lookup time and the cache it uses.
* Lookup Override uses the “Order By” clause by default. SQL Override doesn’t use it and should be manually entered in the query if we require it
* SQL Override can provide any kind of ‘join’ by writing the query
* Lookup Override provides only Non-Equi joins.
* Lookup Override gives only one record even if it finds multiple records for a single condition
* SQL Override doesn’t do that.

**What are the different ways to implement parallel processing in Informatica?**

We can implement parallel processing using various types of partition algorithms:

**Database partitioning:** The Integration Service queries the database system for table partition information. It reads partitioned data from the corresponding nodes in the database.

**Round-Robin Partitioning:** Using this partitioning algorithm, the Integration service distributes data evenly among all partitions. It makes sense to use round-robin partitioning when you need to distribute rows evenly and do not need to group data among partitions.

**Hash Auto-Keys Partitioning:** The Powercenter Server uses a hash function to group rows of data among partitions. When the hash auto-key partition is used, the Integration Service uses all grouped or sorted ports as a compound partition key. You can use hash auto-keys partitioning at or before Rank, Sorter, and unsorted Aggregator transformations to ensure that rows are grouped properly before they enter these transformations.

**Hash User-Keys Partitioning:** Here, the Integration Service uses a hash function to group rows of data among partitions based on a user-defined partition key. You can individually choose the ports that define the partition key.

**Key Range Partitioning:** With this type of partitioning, you can specify one or more ports to form a compound partition key for a source or target. The Integration Service then passes data to each partition depending on the ranges you specify for each port.

**Pass-through Partitioning:** In this type of partitioning, the Integration Service passes all rows from one partition point to the next partition point without redistributing them.

**Explain shared cache and re cache.**

If you configure the lookup as a persistent cache, the server saves the lookup under an anonymous name. Shared cache allows you to use this cache in other mappings by directing it to an existing cache.

**Mention a few design and development best practices for Informatica.**

**Reusability** – in order to react quickly to potential changes, use Informatica components like mapplets, worklets, and reusable transformations.

**Scalability** – when designing and developing mappings, it is a good practice to keep volumes in mind. This is caching, queries, partitioning, initial vs incremental loads.

**Simplicity** – it is recommended to create multiple mappings instead of few complex ones. Use Staging Area and try to keep the processing logic as clear and simple as possible.

**Modularity** – use the modular design technique (common error handling, reprocessing).

Mapping development best practices

**Source Qualifier** – use shortcuts, extract only the necessary data, limit read of columns and rows on source. Try to use the default query options (User Defined Join, Filter) instead of using SQL Query override which may impact database resources and make unable to use partitioning and push-down.

**Expressions** – use local variables to limit the amount of redundant calculations, avoid datatype conversions, reduce invoking external scripts (coding outside of Informatica), provide comments, use operators (||, +, /) instead of functions. Keep in mind that numeric operations are generally faster than string operations.

**Filter** – use the Filter transformation as close to the source as possible. If multiple filters need to be applied, usually it’s more efficient to replace them with Router.

**Aggregator** – use sorted input, also use as early (close to the source) as possible and filter the data before aggregating.

**Joiner** – try to join the data in Source Qualifier wherever possible, and avoid outer joins. It is good practice to use a source with fewer rows, such as a Master source.

## Lookup – relational lookup should only return ports that meet the condition. Call Unconnected Lookup in expression (IIF). Replace large lookup tables with joins whenever possible. Review the database objects and add indexes to database columns when possible. Use Cache Calculator in session to eliminate paging in lookup cache.

<http://www.etltechblog.com/2012/05/tuning-sessions-for-better-performance.html>

[**http://it.toolbox.com/blogs/infosphere/17-mistakes-that-etl-designers-make-with-very-large-data-19264**](http://it.toolbox.com/blogs/infosphere/17-mistakes-that-etl-designers-make-with-very-large-data-19264)

**What if look has million of records?**

Try to use Source qualifier with left join look up table and use push down for SQL.

**What are the differences between Source Qualifier and Joiner Transformation?**

The Source Qualifier can join data originating from the same source database. We can join two or more tables with primary key-foreign key relationships by linking the sources to one Source Qualifier transformation.

If we have a requirement to join the mid-stream or the sources are heterogeneous, then we will have to use the Joiner transformation to join the data.

**Differentiate between joiner and Lookup Transformation.**

Below are the differences between lookup and joiner transformation:

In lookup we can override the query but in joiner we cannot.

In lookup we can provide different types of operators like – “>,<,>=,<=,!=” but, in joiner only “= “ (equal to )operator is available.

In lookup we can restrict the number of rows while reading the relational table using lookup override but, in joiner we cannot restrict the number of rows while reading.

In joiner we can join the tables based on- Normal Join, Master Outer, Detail Outer and Full Outer Join but, in lookup this facility is not available .Lookup behaves like Left Outer Join of database.

**Pushdown Optimization:**

* **Transformations can be pushed**: expression, joiner, filter, aggregator and sorter transformations are pushed to the database.
* **While using look up transformation**, look sup needs to be returning only one value hence select option “Report Error” on multiple matches.
* **Variable ports in Expression** can’t be used in Full Push down.
* Sequence and Sorter can’t be used together.

One can push transformation logic to the source or target database using pushdown optimization. The Integration Service translates the transformation logic into SQL queries and sends the SQL queries to the source or the target database which executes the SQL queries to process the transformations. The amount of transformation logic one can push to the database depends on the database, transformation logic, and mapping and session configuration. The Integration Service analyzes the transformation logic it can push to the database and executes the SQL statement generated against the source or target tables, and it processes any transformation logic that it cannot push to the database.

* source-side pushdown optimization
* target-side pushdown optimization
* full pushdown optimization
  + the source and target must be on the same database.

**Note:**

1. Push Sequence Generator transformation logic to a database, we must configure the session for pushdown optimization with Sequence.
2. To enable the Integration Service to create the view objects in the database we must configure the session for pushdown optimization with View.

**Configuring Parameters for Pushdown Optimization:**

Depending on the database workload, we might want to use source-side, target-side, or full pushdown optimization at different times and for that we can use the **$$PushdownConfig** mapping parameter

* **none** i.e the integration service itself processes all the transformations.
* **Source [Seq View],**
* **Target [Seq View],**
* **Full [Seq View]**

**Informatica pushdown is more effective** when we used DBs like Oracle, SQL Server,Teradata and Sybase compare to ODBC connect drivers. Using native drivers it converts transformations into DB SQL rather than ANSI SQL.

**The Integration Service and databases can produce different results** when processing the same transformation logic. The Integration Service sometimes converts data to a different format when it reads data. The Integration Service and database may also handle null values, case sensitivity, and sort order differently.

* **Nulls treated as the highest or lowest value.** In the session, you configure nulls as the lowest value in the sort order. Oracle treats null values as the highest value in the sort order.
* **Sort Order:** For example, you want to push the transformations in a session to a Microsoft SQL Server database, which is configured to use a sort order that is not case sensitive. You configure the session properties to use the binary sort order, which is case sensitive.
* **Case sensitivity:** the Integration Service uses case sensitive queries and the database does not.
* **Numeric values converted to character values:** The database can convert numeric values to an unacceptable character format. For example, a table contains the number 1234567890. When the Integration Service converts the number to a character value, it inserts the characters ‘1234567890.’ However, a database might convert the number to ‘1.2E9.’
* **Precision.**
* **Current Date and time:** When you push SYSDATE or SYSTIMESTAMP to the database, the database server returns the timestamp in the time zone of the database server, not the Integration Service.

**Error handling:**

**Fatal and Non-fatal errors:**

**How to handle non-fatal errors:**

You can select session properties to handle various levels of errors occurred in mapping.

Error messages can be handled in flat file or DB.

In case if handled in DB, It creates 4 tables automatically with adjacent profile name given in property.

[**https://www.youtube.com/channel/UClhTKfuOnx4uC9VRnHw712Q**](https://www.youtube.com/channel/UClhTKfuOnx4uC9VRnHw712Q)

**Informatica Variables and parameters:**

**Variables can be used in following transformations:**

* Expression
* Filter
* Router
* Update Strategy

**Start Value:**

The start value is the value of the variable at the start of the session. The Integration Service looks for the start value in the following order:

* Value in parameter file
* Value saved in the repository
* Initial value
* Default value

$ - These are the system variables like $Bad file,$inputfile, $output file, $DB connection

$$ - user defined variables

**Informatica Performance Improvement Tips**

* Use Source Qualifier if the Source tables reside in the same schema.
* Make use of Source Qualifier “Filter" Properties if the Source type is Relational.
* If the subsequent sessions are doing lookup on the same table, use persistent cache in the first session. Data remains in the Cache and available for the subsequent session for usage.
* Use flags as integer, as the integer comparison is faster than the string comparison.
* Use tables with lesser number of records as master table for joins.
* While reading from Flat files, define the appropriate data type instead of reading as String and converting.
* Have all Ports that are required connected to Subsequent Transformations else check whether we can remove these ports.
* Use Router transformation in place of multiple Filter transformations.
* Turn off the Verbose Logging while moving the workflows to Production environment.
* For large volume of data drop index before loading and recreate indexes after load.
* For large of volume of records Use Bulk load Increase the commit interval to a higher value large volume of data.
* Set 'Commit on Target' in the sessions.

**Incremental Aggregation: (session Level Property)**

When you run the session for the first time using the incremental aggregation, then integration service process the entire source and stores the data in two file, index and data file. The integration service creates the files in the cache directory specified in the aggregator transformation properties.

Information Data Quality:

**Main components of IDQ:**

* Model Repository Service: Metadata of IDQ
* Data Integration Service: Underlying Mapping Logic
* Analyst Service :Data profiling
* Human Task: Manual Work

**Why IDQ:**

* You can see how data is flowing and see data at any point of time.
* Match and Consolidators transformations for fuzzy matching and removing duplicates.
* Mainly for data cleansing
* Very intuitive and easy for adhoc operations
* Without developed full mapping we can validate data flow
* Instead of having source and targets separately we can have all objects and changes their modes read or write.
* Reusable mappings so we can use multiple ways in powercenter

**Keygen Transformation:**

Fuzzy matching in IDQ:

Add a **Keygen** transformation and generate a value on set of columns.

Use Keygen strategies soundex/NYSIIS for nearly matching names.

**Match Transformation:**

**Labeler Transformation:** Identify the data have any specific symbols or labels and segregate those records (Marks the records and we can route it)

**Standardizer transformation:** Identify the data have any specific symbols or labels and standardize those records which we can pass to output (Change the record value)

**Address Doctor Transformation:** Correct addresses

**Consolidator Transformations:**

* Helps to identify duplicate records and filter it.
* Create unique records or master records

**Match and consolidator to find near duplicates records**

How development works:

* First you develop mappings in IDQ
* Deploy IDQ mappings as powercenter mappings in Powercenter
* Then create sessions and workflows from there.

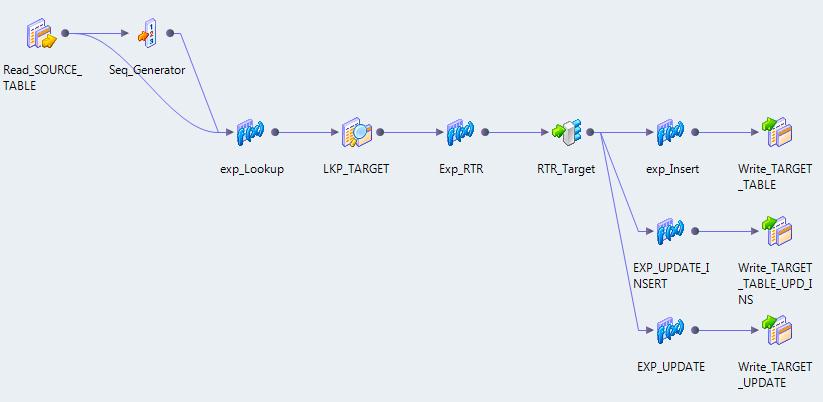
**Migration to Power center from IDQ:**

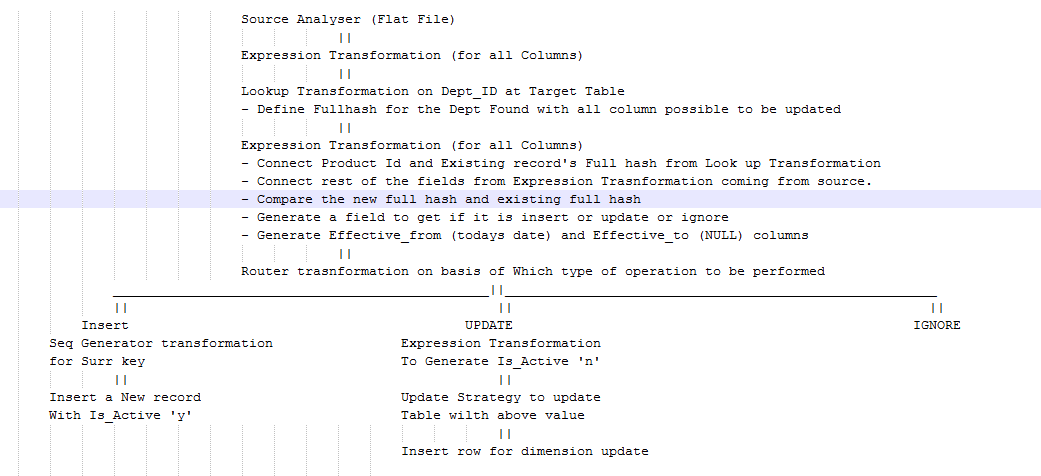
Develop the code in IDQ and move it to powercenter and do the work

Export the objects from file menu to powercenter

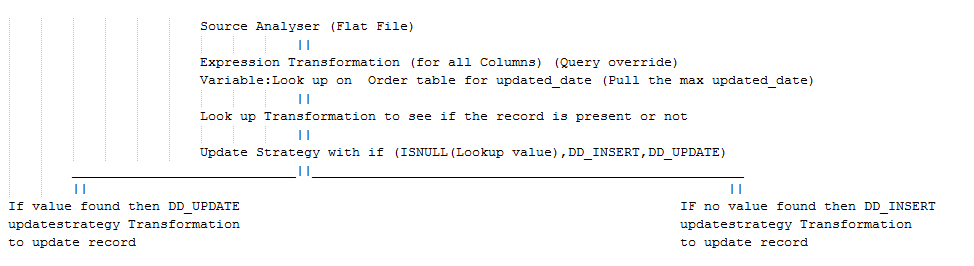
Informatica Scenarios

1. SCD Type 2





1. Incremental Load



References:

<https://www.guru99.com/informatica-interview-questions.html>

[**https://www.guru99.com/normalizer-transformation-informatica.html**](https://www.guru99.com/normalizer-transformation-informatica.html)

[**http://informaticatutorials-naveen.blogspot.in/2011/04/sequence-generator-tf\_20.html**](http://informaticatutorials-naveen.blogspot.in/2011/04/sequence-generator-tf_20.html)

**Rejected Records.**

Target override

Update Strategy

Session Properties

Mapping properties

Debugger

Normalizer transformation

Commit Interval

Handle job which is terminated in between

Security in Informatica

<https://marketplace.informatica.com/solutions/convert_rows_to_columns>

Data Quality Checks:

* Source to target Count testing
* Source to target Data testing
* Data Check and Constraint Testing
* Duplicate Data Check Testing
* Regression Testing
* Incremental Testing
* Threshold Value Validation Testing

|  |  |
| --- | --- |
| Test Scenarios | Test-Cases |
| Structure Validation | It involves validating the source and the target table structure as per the mapping document.  Data type should be validated in the source and the target systems.  The length of data types in the source and the target system should be same.  Data field types and their format should be same in the source and the target system.  Validating the column names in the target system. |
| Validating Mapping document | It involves validating the mapping document to ensure all the information has been provided. The mapping document should have change log, maintain data types, length, transformation rules, etc. |
| Validate Constraints | It involves validating the constraints and ensuring that they are applied on the expected tables. |
| Data Consistency check | It involves checking the misuse of integrity constraints like Foreign Key.  The length and data type of an attribute may vary in different tables, though their definition remains same at the semantic layer. |
| Data Completeness Validation | It involves checking if all the data is loaded to the target system from the source system.  Counting the number of records in the source and the target systems.  Boundary value analysis.  Validating the unique values of primary keys. |
| Data Correctness Validation | It involves validating the values of data in the target system.  Misspelled or inaccurate data is found in table.  Null, Not Unique data is stored when you disable integrity constraint at the time of import. |
| Data Transform validation | It involves creating a spreadsheet of scenarios for input values and expected results and then validating with end-users.  Validating parent-child relationship in the data by creating scenarios.  Using data profiling to compare the range of values in each field.  Validating if the data types in the warehouse are same as mentioned in the data model. |
| Data Quality Validation | It involves performing number check, date check, precision check, data check, Null check, etc.  Example − Date format should be same for all the values. |
| Null Validation | It involves checking the Null values where Not Null is mentioned for that field. |
| Duplicate Validation | It involves validating duplicate values in the target system when data is coming from multiple columns from the source system.  Validating primary keys and other columns if there is any duplicate values as per the business requirement. |
| Date Validation check | Validating date field for various actions performed in ETL process.  Common test-cases to perform Date validation −  From\_Date should not greater than To\_Date  Format of date values should be proper.  Date values should not have any junk values or null values |
| Full Data Validation Minus Query | It involves validating full data set in the source and the target tables by using minus query.  You need to perform both source minus target and target minus source.  If the minus query returns a value, that should be considered as mismatching rows.  You need to match the rows in source and target using the Intersect statement.  The count returned by Intersect should match with the individual counts of source and target tables.  If the minus query returns no rows and the count intersect is less than the source count or the target table count, then the table holds duplicate rows. |
| Other Test Scenarios | Other Test scenarios can be to verify that the extraction process did not extract duplicate data from the source system.  The testing team will maintain a list of SQL statements that are run to validate that no duplicate data have been extracted from the source systems. |
| Data Cleaning | Unwanted data should be removed before loading the data to the staging area. |