



THE UNIVERSITY OF
MELBOURNE

ISYS 90086
Data Warehousing
2020 Summer Semester

Assignment 2 – ETL Process

Rongbing Shan 945388 rashan@student.unimelb.edu.au

Table of Contents

1. Exclusive Summary.....	3
2. Design of the ETL process	4
2.1. Date Dimension table	5
2.2. Agents Dimension table.....	5
2.3. Products Dimension table.....	6
2.4. Customers Dimension table	8
2.5. Transactional Sales fact table	9
3. Design of the Data Warehouse.....	12
3.1. Date_Dimension	12
3.2. Agents	12
3.3. Products	13
3.4. Customers	13
3.5. Transactional_Sales	13
4. Data Dictionary.....	14
4.1. ETL Jobs	14
4.2. File Handling.....	17
Appendix 1 – Work Breakdown.....	18

Word Count: 2382

1. Exclusive Summary

This report aims to state the overall design and implementation of the ETL process for Overhill. Pentaho data integration tool is chosen as the ETL tool for this project. All procedures as well as the issues addressed in the process will be explained in detail in this report.

The data warehouse for Overhill contains two sources and the data are in excel files for processing into our new data warehouse. The data will be validated and then loaded into staging area. After cleansing and transformation, the data will be loaded from staging to the storage area for further reporting. The overall process is illustrated in figure 1.

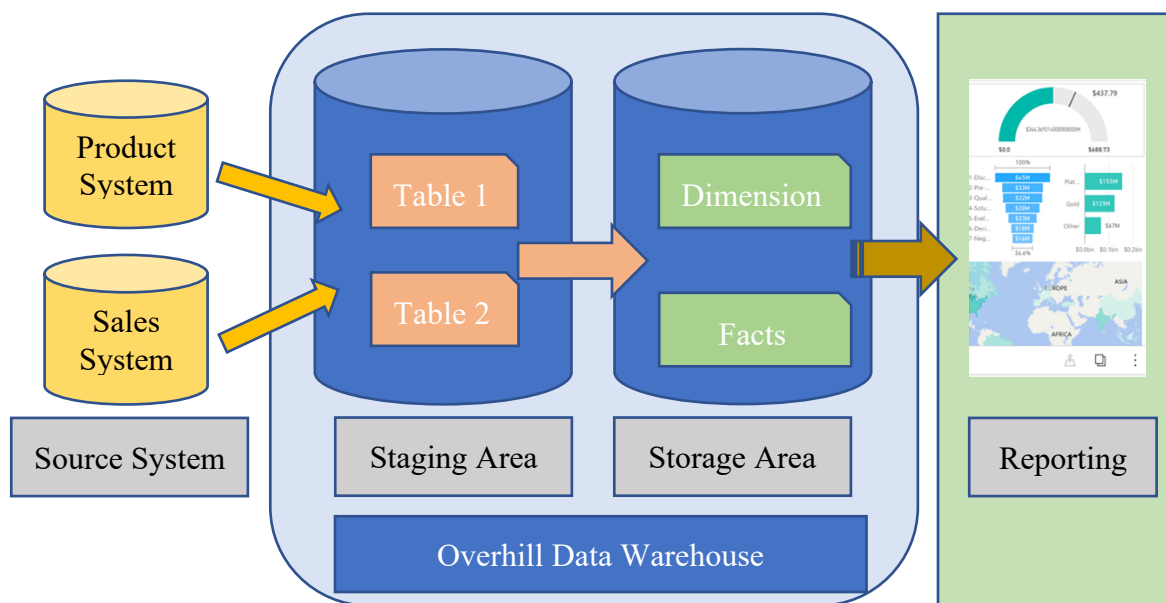


Figure 1. Overall ETL flow

During the ETL process, multiple problems are handled. First issues to address is the data quality problem, the data form and data type can be inconsistent. And the data comes from excel files, which is an unstable source containing data errors like incorrect date formats. Hence, the data quality control is a basic issue in our ETL process. Also, we will need to derive data needed from current values like separating first and last name from whole names. At last, the data warehouse will keep track of the changing history of dimensions like customer information, agent information through Slowly Changing Dimension (SCD), which is also a challenge in the process.

Some re-design on the database is carried out based on the study of source system, which mainly lies in the fact table, to include more information on cost, commission, etc.

2. Design of the ETL process

Pentaho Data Integration is used as the tool for ETL process in this project. In this section, the whole process will be discussed in detail from source file to data warehouse.

In general, for the ETL process, there will be three file folders for loading and error handling, **OLTPData**, **archive**, and **log**. The source data files will be put in **OLTPData** folder for loading, if the files are successfully loaded, they will be moved into **archive** with timestamp added in the filename. When errors on the data is detected, a log file will be generated for those data and be put in the **log** with error information folder for administrators to trace the error.

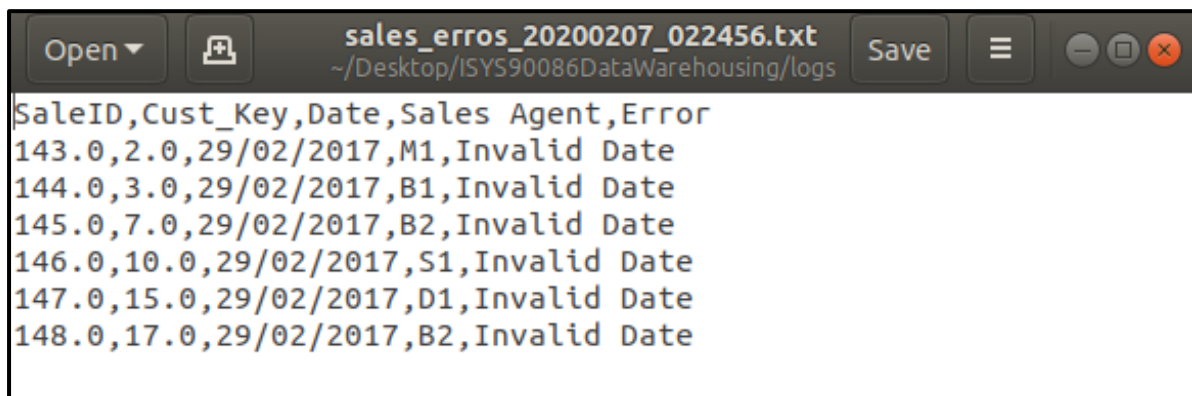


Figure 2. Sample log file with invalid dates

Two databases are created for the project: **Staging** and **Overhill** as illustrated in figure 1. Staging database is used to store staging tables while Overhill will be the main data warehouse for analysis purpose.

Transformations and Jobs will be divided into three types in this project, **Staging**, **Initial Load** and **Incremental Load**. Staging transformations aims to load the data from the file with basic quality control and error handling. Initial transformations run only once to load the data into data warehouse for the first time and incremental load handles further transformations after initial loading. For better explanation, a prefix definition table is given below for reference as in Table 1.

Prefix	Definition
<i>STG_</i>	Staging
<i>INT_</i>	Initial Loading
<i>INC_</i>	Incremental Loading

Table 1. Prefix definition

Next, we will introduce the ETL process by each table in the data warehouse.

2.1. Date Dimension table

Date dimension table will be loaded into the database for only once, so only the initial load will be required for this part of ETL. Also, for the same reason of on time loading, there will be no staging table for date dimension data for simplicity.

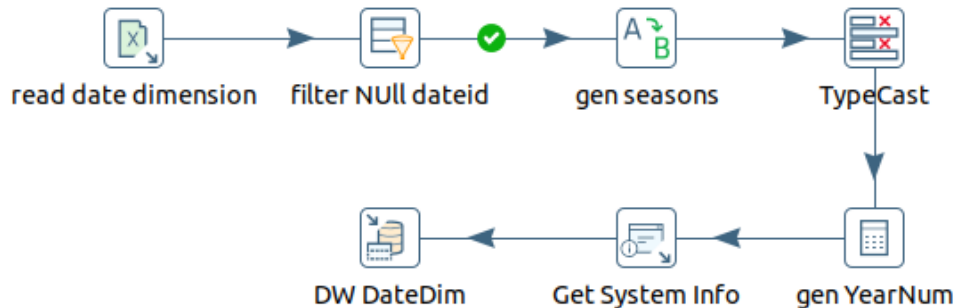


Figure 3. Transformation: ETL_DateDim

The process reads excel source “DimDates.xlsx” with columns as ‘string’ type. Then we will filter ‘Null’ data out. Also, two more data filed will be generated in Date Dimension Table, [Season] and [YearNum]. As current year in the data goes only up to 2039, we will use an insert/update method for loading data into the data warehouse table, so the same ETL task will can be used for loading data after 2039 or fix data on certain day. System timestamp is inserted together with each record. Target table for this transformation is {Overhill.Date_Dimension}.

2.2. Agents Dimension table

Agents dimension will have data first loaded from excel into Staging database and then into Overhill data warehouse.

Staging (STG_Agents)

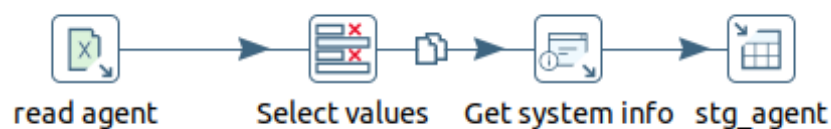


Figure 4. Transformation: STG_Agents

The staging phase reads excel source files “Sales Agent.xlsx” with all columns as ‘string’ type. Then data is cast into the desired data type and truncate/insert into the {Staging.STG_Agents} together with system timestamp.

Initial load (INT_Agents)



Figure 5. Transformation: INT_Agents

For initial load, will be {Staging.STG_Agents} and target is {Overhill.Agents}. Agent name will be split into first name and last name. Also, for initial loading, the SCD2 files [Version], [StartDate] and [EndDate] will be inserted manually together with system timestamps. The data will be truncate/insert into the target table.

Incremental load (INC_Agents)



Figure 6. Transformation: INC_Agents

For incremental load, it is similar to initial load except for that the SCD type 2 fields will be handled by Pentaho Kettle components.

2.3. Products Dimension table

Products dimension will have data first loaded from excel into Staging database and then into Overhill data warehouse.

Staging (STG_Products)

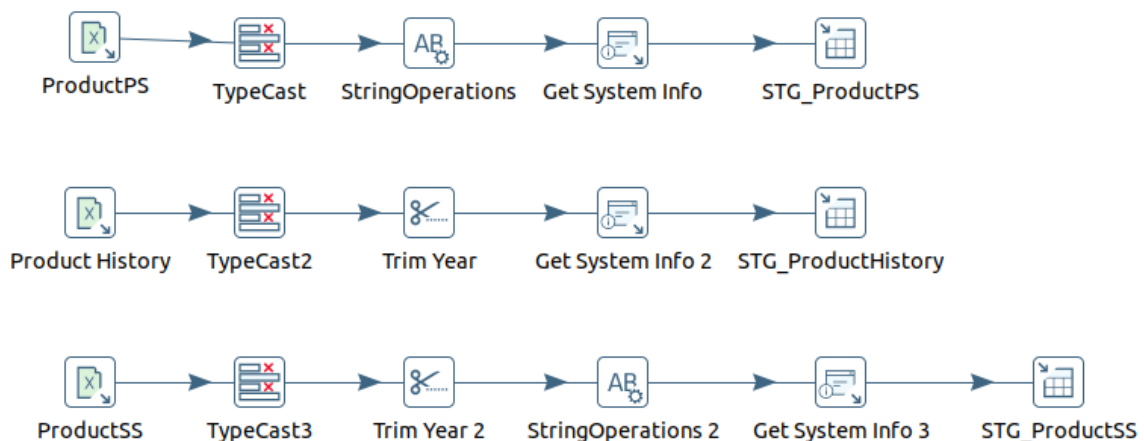


Figure 7. Transformation: STG_Products

The staging phase reads three excels source files “Product.xlsx”, ”Product History.xlsx” from Production System and “Product.xlsx” from Sales System, with all columns as ‘string’ type.

Then data is cast into the desired data type and strings goes through “trim” operation for better data quality. Then the data is separately truncate/insert into three tables {Staging.STG_ProductPS}, {Staging.STG_ProductHistory} and {Staging.STG_ProductSS} together with system timestamp for each record.

Initial load (INT Products)

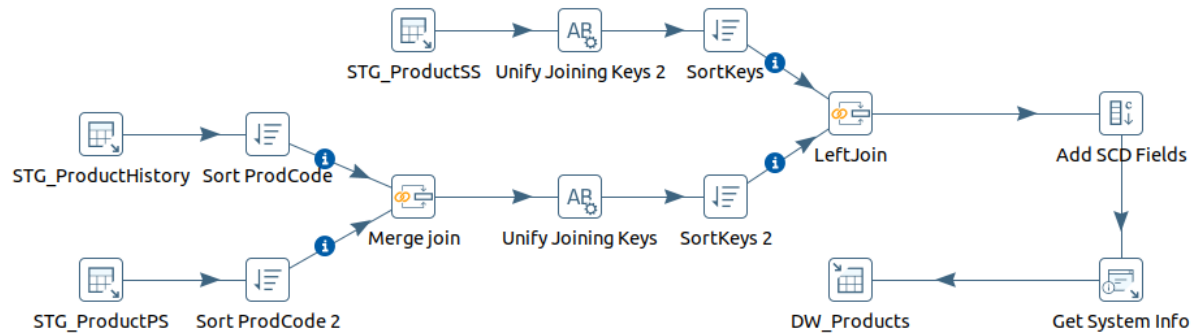


Figure 8. Transformation: INT_Products

For initial load, will be {Staging.STG_ProductPS} {Staging.STG_ProductHistory} and {Staging.STG_ProductSS} and target is {Overhill.Products}. The two data sources from Production system {STG_ProductPS} {STG_ProductHistory} are first joined together by their [ProdCode] after which is sorted. Then {STG_ProductSS} is joined with the result table though a left join where {STG_ProductSS} is the left table. The joining keys will be product description, group and produced years, and all of these keys are “trimmed” and changed to upper case before joining. Then SCD2 fields are added manually together with system timestamp, then the data truncate/insert into the target table.

Incremental load (INC Products)

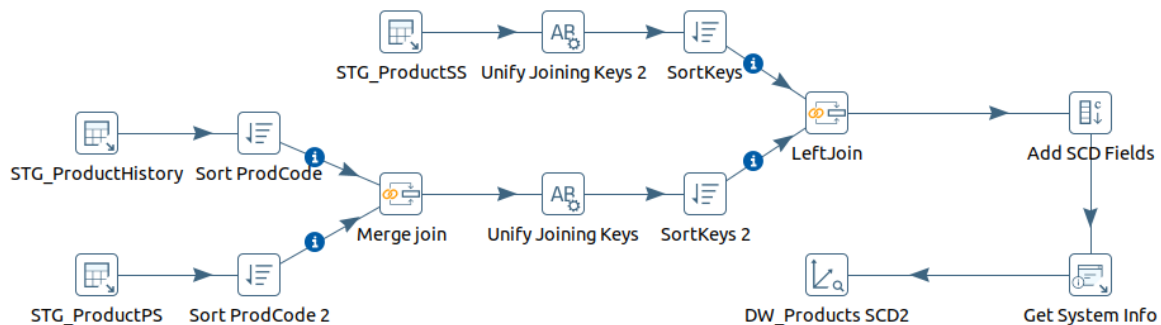


Figure 9. Transformation: INC_Products

For incremental load, it is similar to initial load except for that the SCD type 2 fields will be handled by Pentaho Kettle components.

2.4. Customers Dimension table

Customers dimension will have data first loaded from excel into Staging database and then into Overhill data warehouse.

Staging (STG Market & STG Customers)



Figure 10. Transformation: STG_Market



Figure 11. Transformation: STG_Customers

For customers, there will be two separating staging transformations, for market and customers. That's because the market data is assumed to have a much lower updated frequency than the customer data. By separating the two transformations, we can still access to staging data of markets from last update for new customer updates, which delivers more flexibility in ETL job schedules and administrations.

The Market staging will read excel source file "Market.xlsx" with all columns as string. For better joining maintenance, we change the market ID to upper case and truncate/insert the data into {Staging.STG_Market}.

The Customers staging phase will read excel source files "Customer *** *.xlsx" with wildcard search with all columns as 'string' type. Then data is cast into the desired data type and strings are trimmed, market ID is changed to upper case. Also, before inserting, we sort the data by CustID for better data representations. Then we truncate/insert into the {Staging.STG_Customers} together with system timestamp for each record.

Initial load (INT Customers)

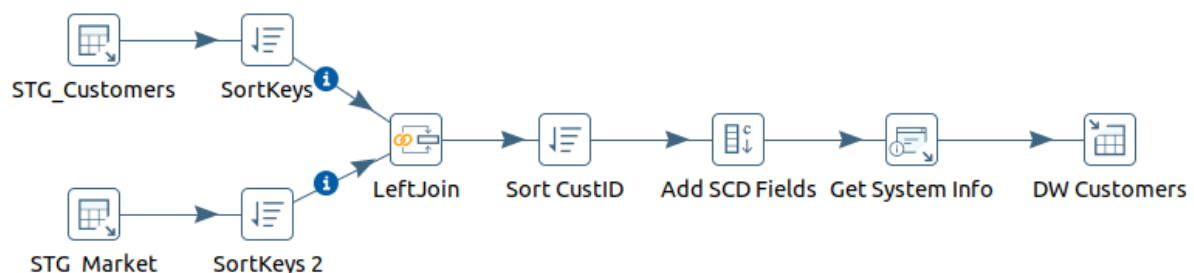


Figure 12. Transformation: INT_Customers

For initial load, will be {Staging.STG_Customers} {Staging.STG_Market} and target is {Overhill.Customers}. The customer table will left join with market table by sorted Market ID to get the descriptions of each market. Then the data is sorted again by CustID. Also, for initial loading, the SCD2 files [Version], [StartDate] and [EndDate] will be inserted manually together with system timestamps. The data will be truncate/insert into the target table.

Incremental load (INC_Customers)

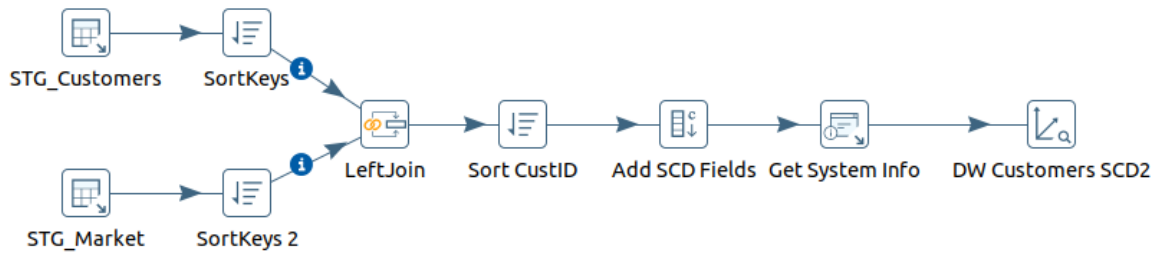


Figure 13. Transformation: INC_Customers

For incremental load, it is similar to initial load except for that the SCD type 2 fields will be handled by Pentaho Kettle components.

2.5. Transactional Sales fact table

Sales fact will have data first loaded from excel into Staging database and then into Overhill data warehouse. It should be run only after all other dimension transformations are finished since there is foreign key constraints checking.

Staging (STG_Sales)

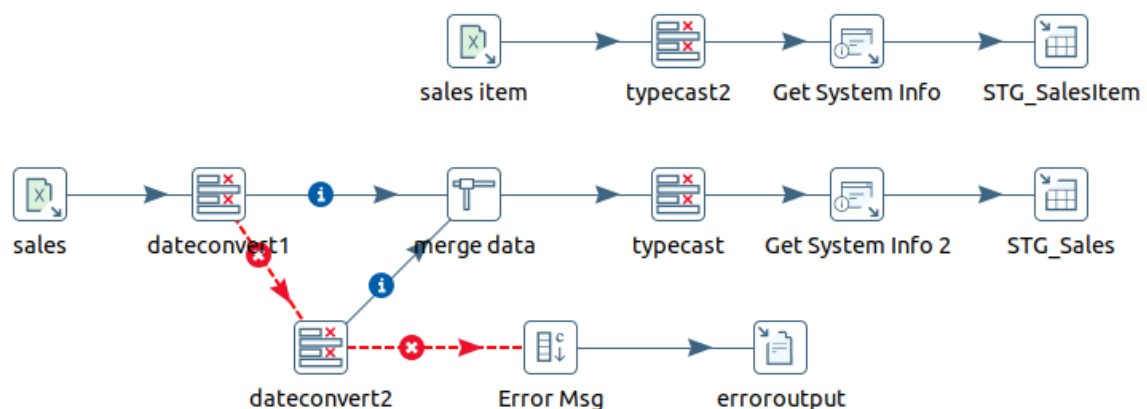


Figure 14. Transformation: STG_Sales

The staging phase will read excels source files “Sale.xlsx” and “SalesItem.xlsx” and all columns will be loaded as ‘string’ type.

For “SalesItem.xlsx”, the data will be cast into desired types and then truncate/insert into {Staging.STG_SalesItem} with system timestamp.

For “Sales.xlsx”, the data will first goes into two date conversions, to convert the mixed strings of different date format (yyyy-MM-dd and dd/MM/yyyy) into unifying date formats. If there still remains record that cannot be cast into correct date type, they will be written into the {log} folder with error messages, and the file name would be the source filename added with date and time. For the records with no date format issue, they will be cast into desired types and truncate/insert into {Staging.STG_Sales} with system timestamp.

Initial load (INT_Sales)

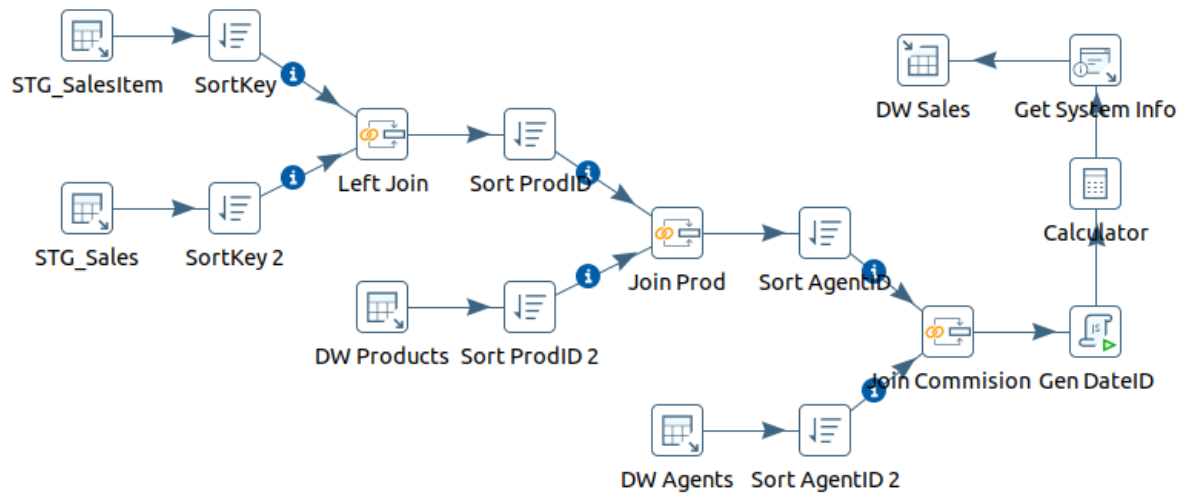


Figure 15. Transformation: INT_Sales

For initial load, will be {Staging.STG_Sales}, {Staging.STG_SalesItem}, {Overhill.Products}, {Overhill.Agents} and target is {Overhill.Transaction_Sales}. Sales Item data will left join with Sales data by sorted [SalesID] to get a transactional sales data. Then the result data is joined with {Overhill.Products} through sorted [ProdID] to get product cost, labeled price. Also, the result data will be joined with {Overhill.Agents} through sorted [AgentID] to get commission rates. Then we will convert the original [Date] field who is in date type into [DateID] who is of Integer type. Next, the desired statistical results will be calculated as shown in figure 16. Then we will truncate/insert the data into the target table together with system timestamp.

Step name

Calculator

☒ Throw an error on non existing files

Fields:

	New field	Calculation	Field A	Field B
1	TotalSales	A * B	UnitSales	UnitPrice
2	TotalCost	A * B	UnitSales	UnitCost
3	TotalMargin	A - B	TotalSales	TotalCost
4	AgentCommission	A * B	TotalSales	CommisionRate

Figure 16. Calculator component in INT_Sales

Incremental load (INC Sales)

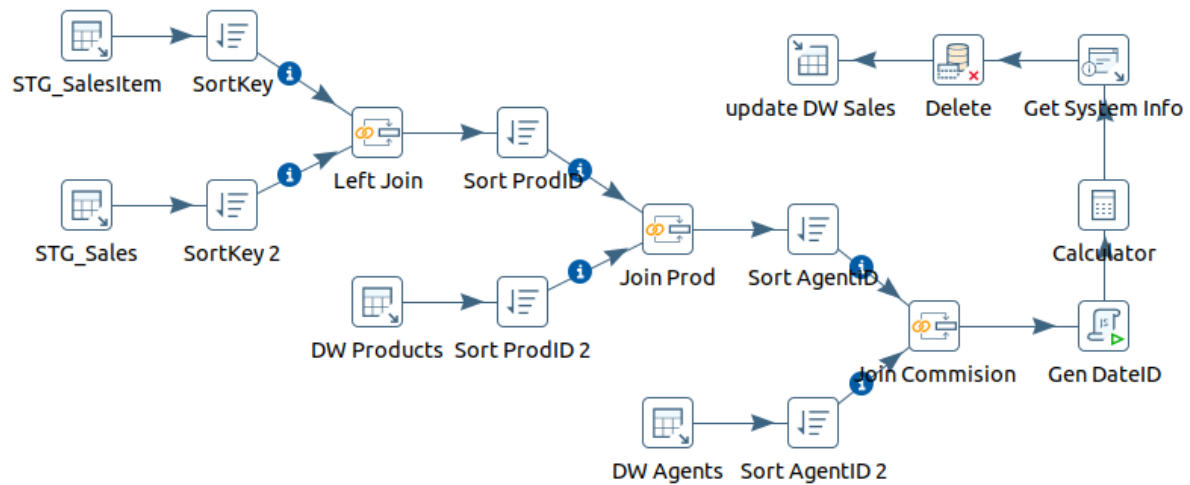


Figure 17. Transformation: INC_Sales

For incremental load, it is similar to initial load except for that we will use a different update method. Here we will use delete/insert for the fact table, because the fact table is on transactional level, which means it will be a very big table with large data volume. Compared with insert/update, delete/update will be faster in performance. Besides, if some extreme error like duplicates records, delete/insert will fix it without manual work.

3. Design of the Data Warehouse

There are some changes made on the design for the data warehouse at this stage of the project. The new schema is shown in figure 18 and original schema is shown in figure 19.

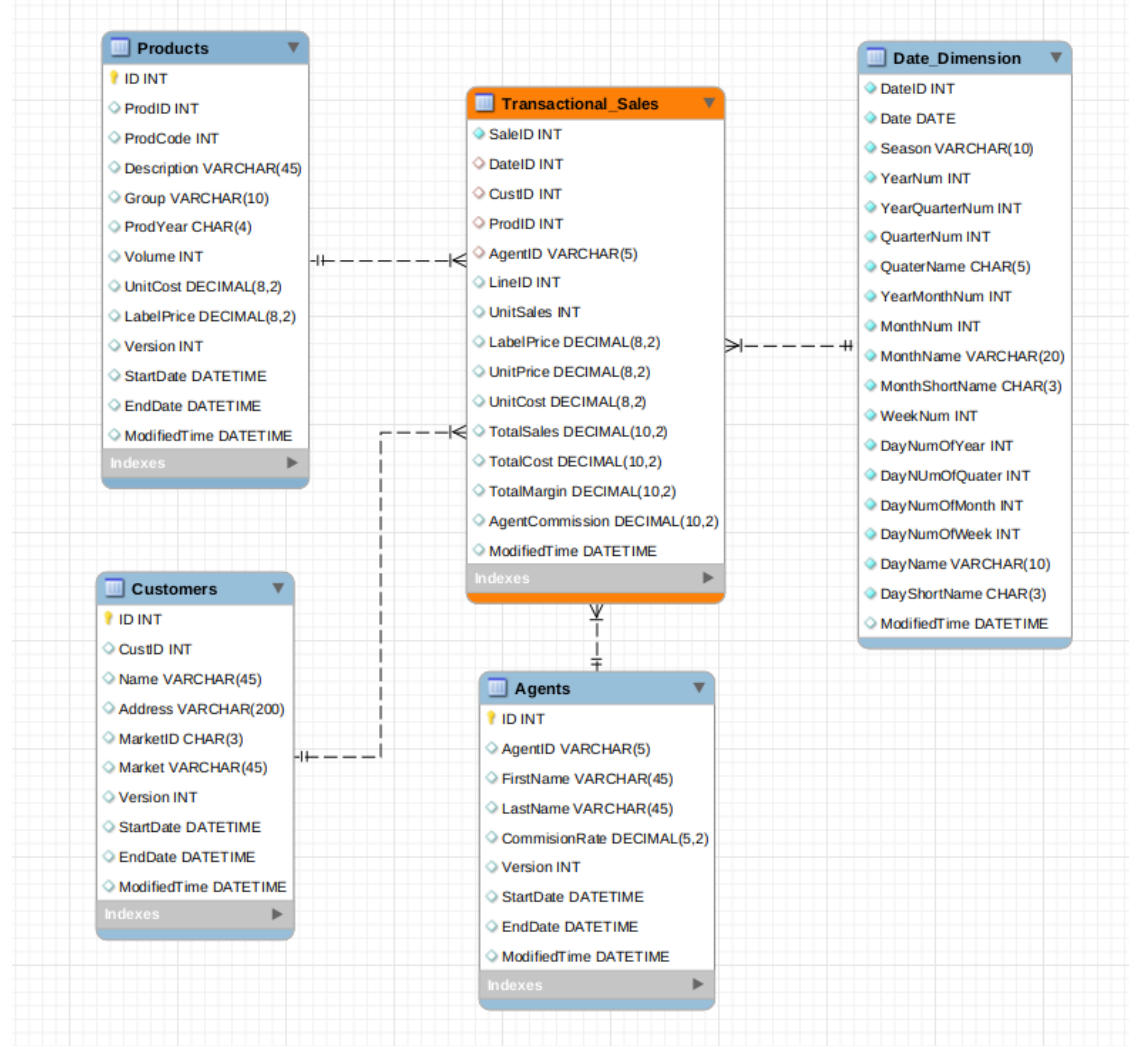


Figure 18. Redesign of database schema

In general, tables are renamed to align the naming syntax for all tables, also to provide more information just from the table name, like “Transactional_Sales” will also indicate the grain for the table. Also, a field [ModifiedTime] is added to all tables for indication of the modified timestamp of the record, for better ETL job maintenance purpose.

3.1. Date_Dimension

The table is renamed from “DATEDIM” to “Date_Dimension” who now has 19 fields instead of 9. Based on the source date dimension file, more fields are added to provide a more comprehensive information on date related data.

3.2. Agents

The table is renamed from “AGENTS” to “Agents”. Besides, the agent name is now splitted into two fields, first name and last name for better querying of analysis purpose. [Flag] field is removed as unnecessary and [Version] is added to keep track of versions of history.

[StartDate] and [EndDate] are changed from INT type to DATETIME type as they should be date fields.

3.3. Products

The table is renamed from “PRODUCTIONS” to “Products”. Also, it is changed to SCD type 2 to keep track of the history, therefore three fields [Version], [StartDate] and [EndDate] are added. Also, the ProdID fields comes from the SalesSystem instead of Production System (as original data dictionary states) as the two source systems use different IDs after studying the source data.

3.4. Customers

The table is renamed from “CUSTOMERS” to “Customers”. Also, the field containing market ID is now named to [MarketID] and the description for market is named as [Market]. [Flag] field is removed as unnecessary and [Version] is added to keep track of versions of history. [StartDate] and [EndDate] are changed from INT type to DATETIME type as they should be date fields.

3.5. Transactional_Sales

The table is renamed from “SALES” to “Transactional_Sales” to indicate the level of detail. Originally, the table contains little information so it would be complicated to do queries for analysis. So more fields are added including [UnitCost], [TotalSales], [TotalCost], [TotalMargin] and [AgentCommission] for quicker analysis purpose.

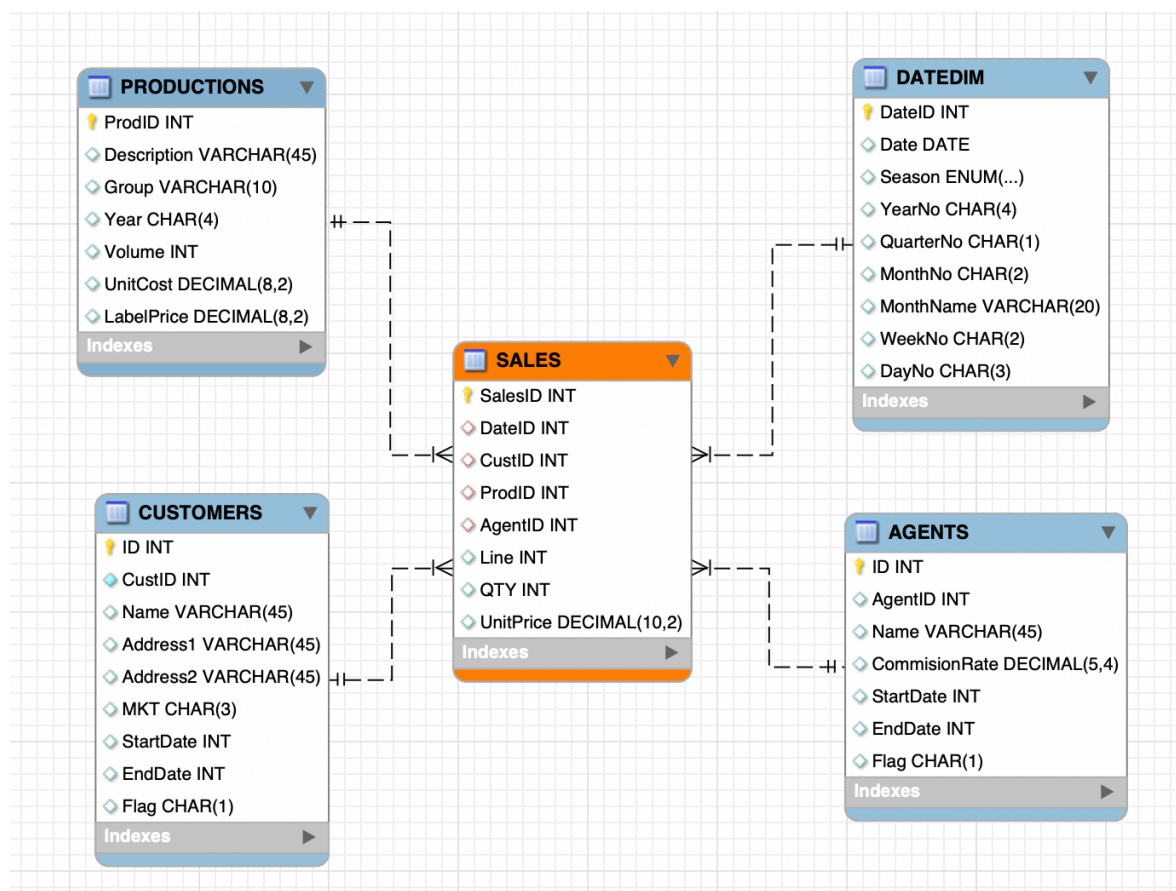
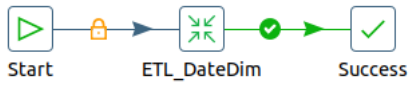


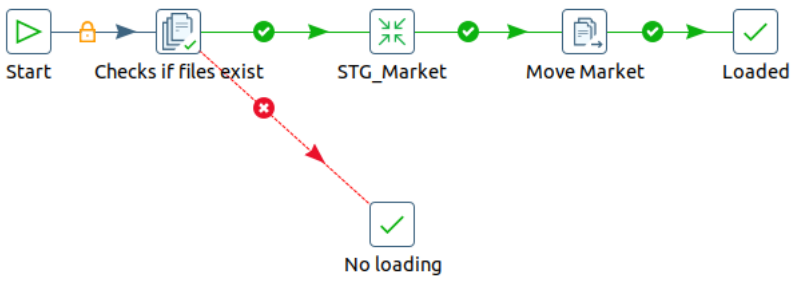
Figure 19. Original Schema

4. Data Dictionary

4.1. ETL Jobs

Most of the ETL jobs are running daily, but the sources files can be provided at their own frequency. For example, if Market information is updated every year, the source file should be provided in ||OLTPData|| folder for input yearly, but the ETL job will check in daily basis if the file is provided. If it is provided it will be loaded, else previous information on Market will be used.

Job Name	INT_DateDim
Job Purpose	To load the date dimension table
Source Tables/Files	“DimDates.xlsx”
Target Tables/Files	Overhill.Date_Dimension
Predecessor	None
Successor	STG_Market
Frequency	Once
	

Job Name	STG_Market
Job Purpose	To load the market information into staging database
Source Tables/Files	“Market.xlsx”
Target Tables/Files	Staging.STG_Market
Predecessor	None / INT_DateDim
Successor	INT_Customers / INC_Customers
Frequency	Daily
	

Job Name	INT_Customers
Job Purpose	Initial loading for customers
Source Tables/Files	“Customer XXX XXXX.xlsx”
Target Tables/Files	Overhill.Customers
Predecessor	INT_DateDim
Successor	INT_Products
Frequency	Once
<pre> graph LR Start[Start] --> Lock[] Lock --> CustomerExists[Customer exists] CustomerExists --> STG_Customers[STG_Customers] STG_Customers --> MoveCustomers[Move Customers] MoveCustomers --> INT_Customers[INT_Customers] INT_Customers --> Success[Success] </pre>	

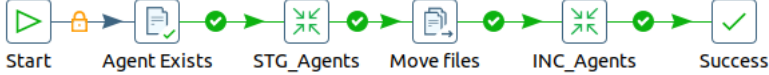
Job Name	INT_Products
Job Purpose	Initial loading for product information
Source Tables/Files	“Product.xlsx”, “Product History.xlsx”, “Product.xlsx”
Target Tables/Files	Overhill.Products
Predecessor	INT_Custoemers
Successor	INT_Agents
Frequency	Once
<pre> graph LR Start[Start] --> Lock[] Lock --> ProductPS[ProductPS] ProductPS --> ProductHistory[ProductHistory] ProductHistory --> ProductSS[ProductSS] ProductSS --> INT_Products[INT_Products] INT_Products --> MoveFiles[Move files] MoveFiles --> Success[Success] </pre>	


Job Name	INT_Agents
Job Purpose	Initial loading for agent data
Source Tables/Files	“Sales Agent.xlsx”
Target Tables/Files	Overhill.Agents
Predecessor	INT_Products
Successor	INT_Sales
Frequency	Once
<pre> graph LR Start[Start] --> Lock[] Lock --> AgentExists[Agent Exists] AgentExists --> STG_Agents[STG_Agents] STG_Agents --> MoveFiles[Move files] MoveFiles --> INT_Agents[INT_Agents] INT_Agents --> Success[Success] </pre>	

Job Name	INT_Sales
Job Purpose	Initial loading for sales data
Source Tables/Files	"Sale.xlsx", "SaleItem.xlsx"
Target Tables/Files	Overhill.Transactiona_Sales
Predecessor	INT_Agents
Successor	None
Frequency	Once
<pre> graph LR Start[Start] --> Lock[Lock] Lock --> SalesExists[Sales exists] SalesExists --> SalesItemExists[Salesitem exists] SalesItemExists --> INT_Sales[INT_Sales] INT_Sales --> MoveFiles[Move files] MoveFiles --> Success[Success] </pre>	

Job Name	INC_Customers
Job Purpose	Incremental load for customer information
Source Tables/Files	"Customer XXX XXXX.xlsx"
Target Tables/Files	Overhill.Customers
Predecessor	STG_Market
Successor	INC_Products
Frequency	Daily
<pre> graph LR Start[Start] --> Lock[Lock] Lock --> CheckFolder[Check if a folder is empty] CheckFolder --> STG_Customers[STG_Customers] STG_Customers --> MoveFiles[Move files] MoveFiles --> INC_Customers[INC_Customers] INC_Customers --> Success[Success] </pre>	

Job Name	INC_Products
Job Purpose	Incremental load for products information
Source Tables/Files	"Product.xlsx", "Product History.xlsx", "Product.xlsx"
Target Tables/Files	Overhill.Products
Predecessor	INC_Custoemers
Successor	INC_Agents
Frequency	Daily
<pre> graph LR Start[Start] --> Lock[Lock] Lock --> ProductPS[ProductPS] ProductPS --> ProductHistory[ProductHistory] ProductHistory --> ProductSS[ProductSS] ProductSS --> INC_Products[INC_Products] INC_Products --> MoveFiles[Move files] MoveFiles --> Success[Success] </pre>	

Job Name	INC_Agents
Job Purpose	Incremental load for agents information
Source Tables/Files	“Sales Agent.xlsx”
Target Tables/Files	Overhill.Agents
Predecessor	INC_Products
Successor	INC_Sales
Frequency	Daily
	

Job Name	INC_Agents
Job Purpose	Incremental load for sales information
Source Tables/Files	“Sale.xlsx”, “SaleItem.xlsx”
Target Tables/Files	Overhill.Transaction_Sales
Predecessor	INC_Agents
Successor	None
Frequency	Daily
	

4.2.File Handling

Source files will be input from **||OLTPData||** folder.

Errors will be input into **||log||** folder with date time.

Loaded source files will be moved to **||archive||** folder with date time.

Appendix 1 – Work Breakdown

The assignment 2 is done by Rongbing Shan for all parts with permission from Humza.