[Introduction to Staging Layer in Data Warehouse 2](#_Toc190609133)

[What is a Staging Layer in Data Warehousing? 2](#_Toc190609134)

[Importance of the Staging Layer in ETL Process 2](#_Toc190609135)

[Staging Layer Workflow in This Project 3](#_Toc190609136)

[Data Flow in the Staging Layer 3](#_Toc190609137)

[Steps in Staging Layer Implementation 3](#_Toc190609138)

[Staging Layer Mapping Example in Informatica 3](#_Toc190609139)

[Outcome and Benefits 4](#_Toc190609140)

[ETL Mapping Documentation (STAGING LAYER) 5](#_Toc190609141)

[ORDER TABLE 5](#_Toc190609142)

[FEEDBACK TABLE 9](#_Toc190609143)

[ORDER ITEM TABLE 13](#_Toc190609144)

[PAYMENT TABLE 16](#_Toc190609145)

[PRODUCTS TABLE 19](#_Toc190609146)

[SELLER TABLE 21](#_Toc190609147)

[USER TABLE 24](#_Toc190609148)

[Introduction to the Data Warehouse Layer 28](#_Toc190609149)

[Key Characteristics of the Data Warehouse Layer 28](#_Toc190609150)

[Data Warehouse Architecture in ETL 28](#_Toc190609151)

[Data Warehouse Layer in My Project 28](#_Toc190609152)

[Data Warehouse Components in My Project 29](#_Toc190609153)

[ETL Process for My Data Warehouse 29](#_Toc190609154)

[Why Data Integrity is Critical in My Project? 29](#_Toc190609155)

[ETL Mapping Documentation (Data Warehouse Layer) 30](#_Toc190609156)

[FEEDBACK TABLE 30](#_Toc190609157)

[PRODUCTS TABLE 37](#_Toc190609158)

[ORDER TABLE 42](#_Toc190609159)

[PAYMENT TABLE 49](#_Toc190609160)

[USER TABLE 56](#_Toc190609161)

[ORDER ITEM TABLE 63](#_Toc190609162)

[SELLER TABLE 69](#_Toc190609163)

[Date Dimension and Time Dimension in Data Warehousing 75](#_Toc190609164)

# Introduction to Staging Layer in Data Warehouse

## What is a Staging Layer in Data Warehousing?

The **staging layer** in a Data Warehouse serves as an intermediate storage area where raw data from multiple sources is temporarily stored before being transformed and loaded into the **Data Warehouse layer**. This layer acts as a buffer between the source systems and the Data Warehouse, ensuring **data cleansing, validation, and transformation** before final processing.

## Importance of the Staging Layer in ETL Process

* **Data Cleansing:** Ensures the data is formatted correctly, removing inconsistencies.
* **Data Validation:** Validates data before it is loaded into the Data Warehouse.
* **Incremental Loading:** Helps in managing new and updated records efficiently.
* **Error Handling:** Any rejected records or errors can be logged without affecting the Data Warehouse.
* **Data Integration:** Combines data from multiple sources into a unified format before final processing.

## Staging Layer Workflow in This Project

In this project, the **staging layer** was designed to load data from **flat files** into temporary staging tables in **Oracle** before transforming and loading it into the Data Warehouse. The ETL process was implemented using **Informatica PowerCenter**, ensuring smooth data movement from sources to targets.

## Data Flow in the Staging Layer

The following diagram illustrates the data flow in the staging layer:

+----------------+ Extract +----------------+ Load +------------------+

| Flat Files | ------------> | Staging Tables | ---------> | Data |

| (Source Data) | | (Temporary DB) | | (Final Target) |

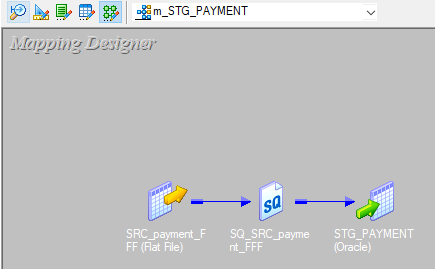
+----------------+ +----------------+ +------------------+

## Steps in Staging Layer Implementation

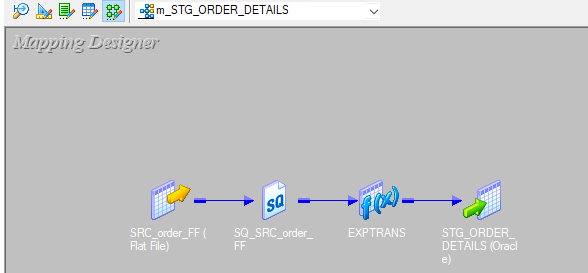
1. **Data Extraction**
   * Data was extracted from seven flat files containing raw e-commerce transactions.
   * Informatica mappings were created to read the flat file data.
2. **Data Loading to Staging Tables**
   * Each flat file was mapped to a corresponding **staging table** in Oracle.
   * No transformations were applied at this stage, ensuring a raw replica of the source.
3. **Data Transformation**
   * Data quality checks were performed (e.g., handling null values, removing duplicates).
   * Basic formatting (e.g., date and number conversions) was applied.
4. **Error Handling & Logging**
   * Invalid records were redirected to **rejection tables** for further analysis.
   * Logging mechanisms were implemented to track failed records.

## Staging Layer Mapping Example in Informatica

For each file, an **Informatica mapping** was created, reading from the source flat file and loading into an **Oracle staging table**. Below is a sample staging mapping:



-NO EXPRESSION NEEDED



-EXPRESSION TRANSFORMATION NEED , BECAUSE ORDER FLAT FILE CONTAINS DATE AND SOURCE IS FLAT FILE SO DATE IS STRING IN THAT CASE BUT THE TARGET IS ORACLE DATABASE SO WE NEED TO CONVERT IT TO

## Outcome and Benefits

* **Ensured Data Integrity:** Staging tables captured data as-is, preventing corruption of final warehouse data.
* **Improved Performance:** Reduced load on the Data Warehouse by pre-processing data in the staging layer.
* **Scalability:** Allowed incremental data loads for future business needs.

The **staging layer** was an essential part of the ETL process, ensuring that only **validated, cleansed, and formatted data** was loaded into the **Data Warehouse** for analytics and reporting.

# ETL Mapping Documentation (STAGING LAYER)

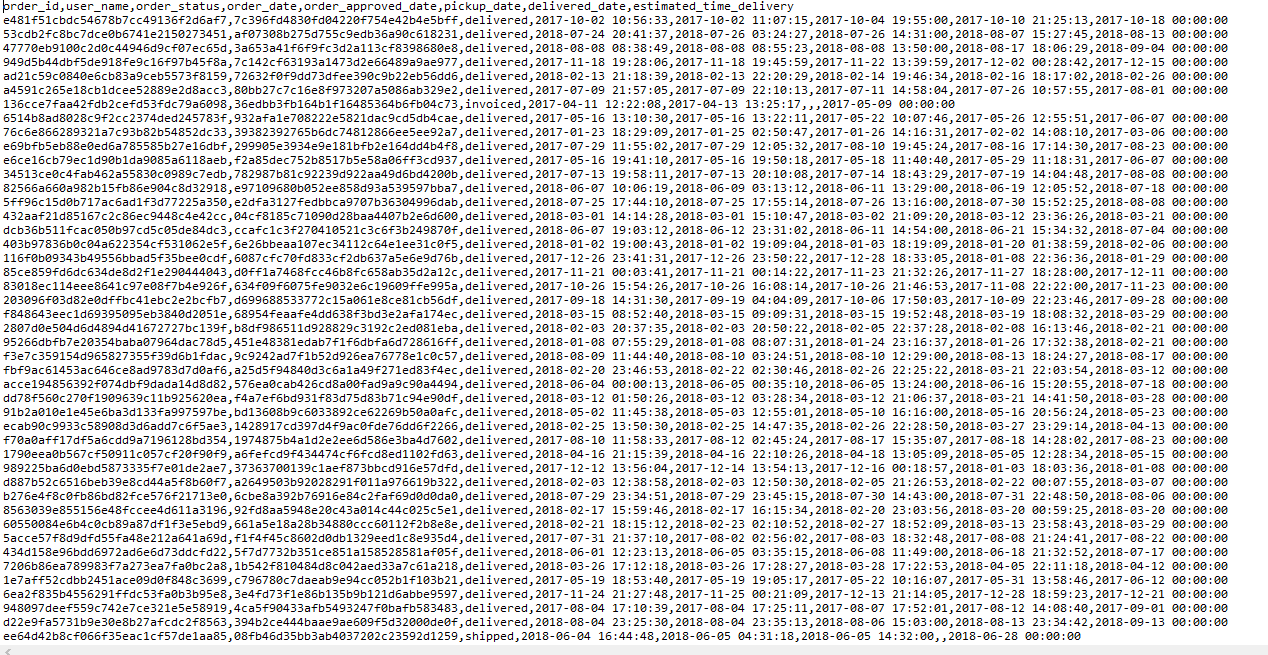
## ORDER TABLE

### 1. Mapping Overview (ORDER)

* **Mapping Name:** [m\_STG\_ORDER\_DETAILS]
* **Source File:** [SRC\_order\_FF)]
* **Target Table:** [STG\_ORDER\_DETAILS]
* **Description:**  
  [DATA IS LOADED FROM SOURCE FILE TO STAGING TABLE]

### 2. Source File Details

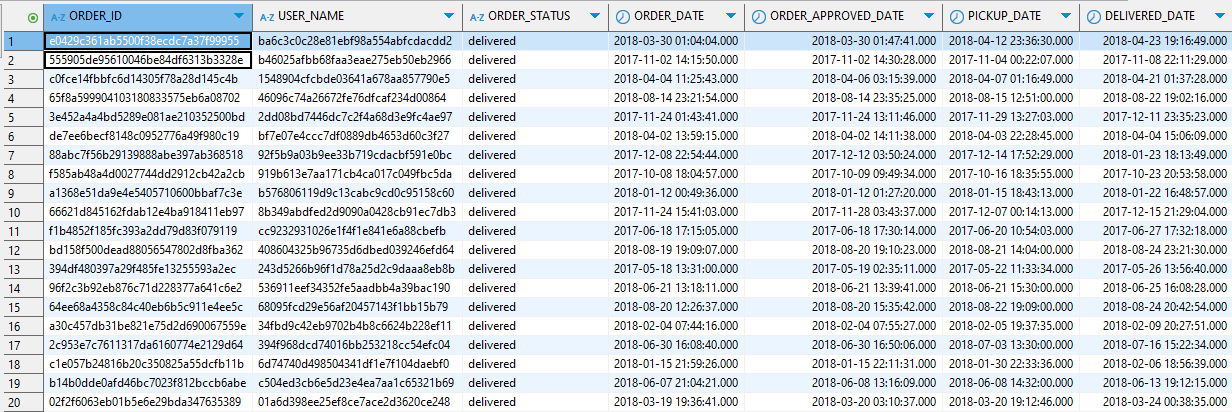
* **File Name:** [SRC\_order\_FF]
* **File Type:** [CSV]
* **Delimiter:** [Comma]

**Column Structure**

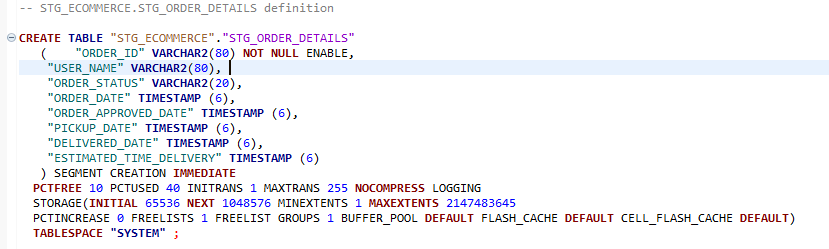
### 3. Target (Staging Table) Details

* **Table Name:** [STG\_ORDER\_DETAILS]

**Column Structure**

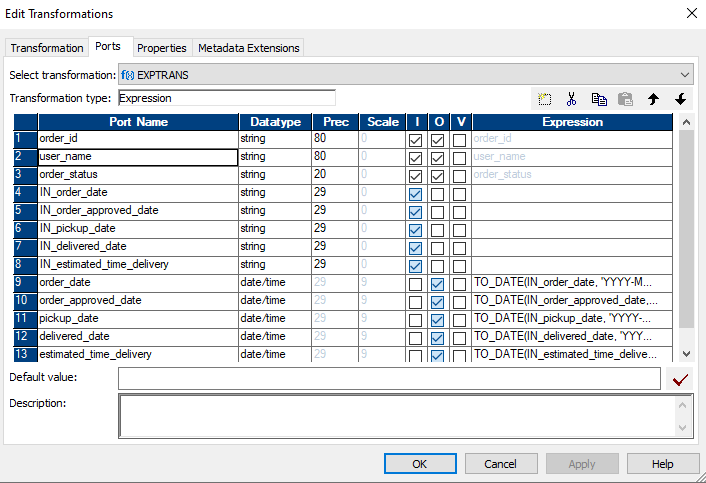


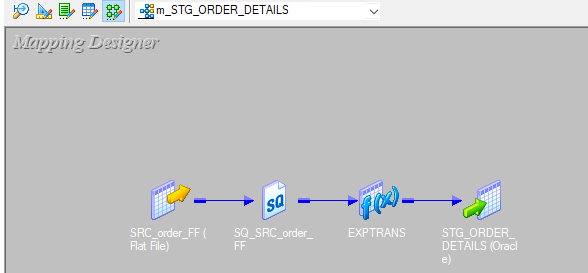
**DDL**

****

### 4. ETL Mapping

* **Expression Transformation:**



* **Screenshot:** 

## FEEDBACK TABLE

### 1. Mapping Overview(FEEDBACK)

* **Mapping Name:** [m\_STG\_FEEDBACK]
* **Source File:** [SRC\_feedback\_FFF]
* **Target Table:** [STG\_FEEDBACK]
* **Description:**  
  [DATA IS LOADED FROM SOURCE TO STAGING TABLE]

### 2. Source File Details

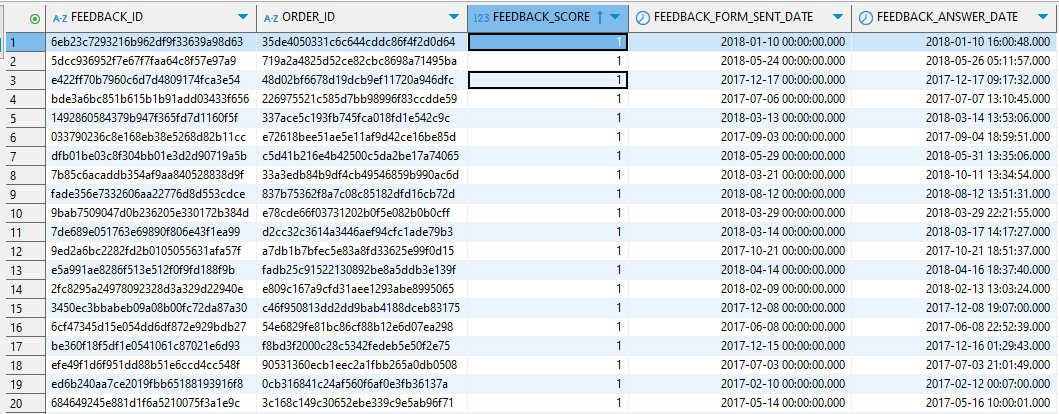
* **File Name:** [SRC\_feedback\_FFF]
* **File Type:** [CSV]
* **Delimiter:** [Comma]

**Column Structure**

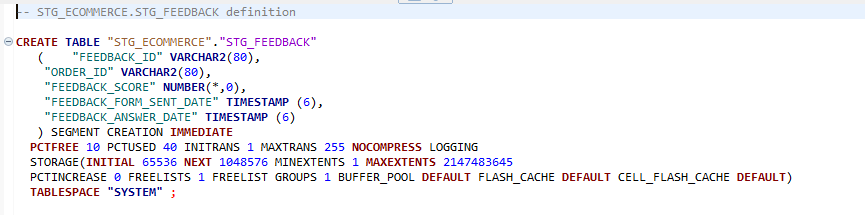
****

### 3. Target (Staging Table) Details

* **Table Name:** [STG\_FEEDBACK]

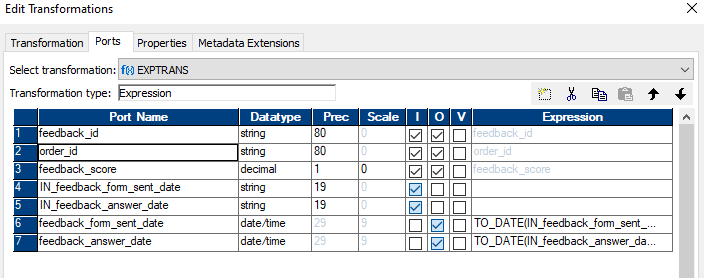
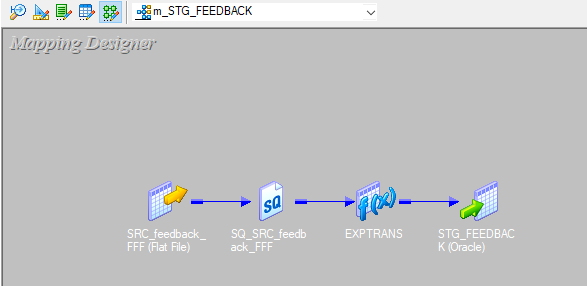
**Column Structure**

**DDL**

****

### 4. ETL Mapping

**Transformations Applied:**

* **Expression Transformation:** 
* **Screenshot:** 

## ORDER ITEM TABLE

### 1. Mapping Overview(ORDER\_ITEM)

* **Mapping Name:** [m\_STG\_ORDER\_ITEM\_DETAILS]
* **Source File:** [SRC\_order\_item\_FFF]
* **Target Table:** [STG\_ORDER\_ITEM\_DETAILS]
* **Description:**[DATA IS LOADED FROM SOURCE TO STAGING TABLE]

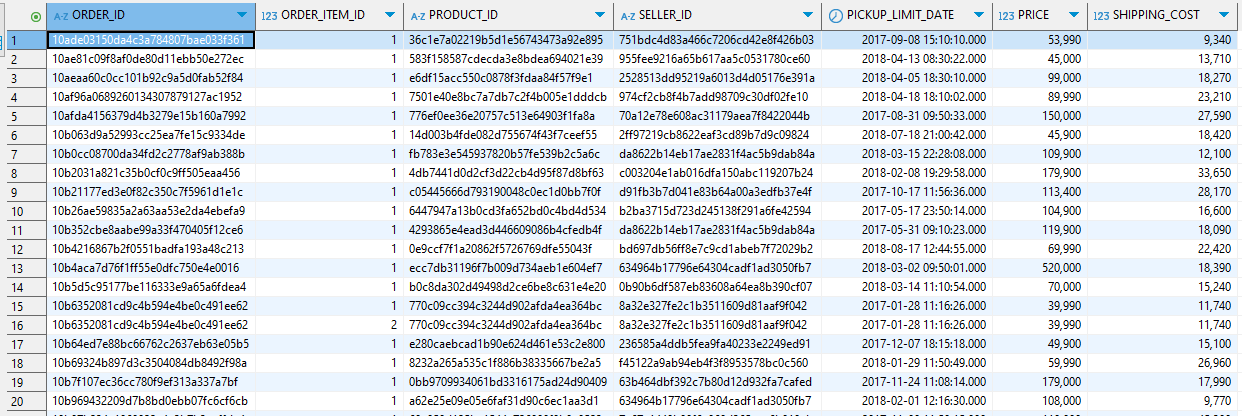
### 2. Source File Details

* **File Name:** [SRC\_order\_item\_FFF]
* **File Type:** [CSV]
* **Delimiter:** [Comma]

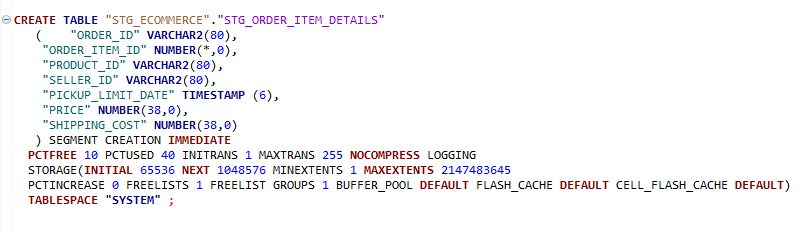
**Column Structure**

### 3. Target (Staging Table) Details

* **Table Name:** [STG\_ORDER\_ITEM\_DETAILS]

**Column Structure**

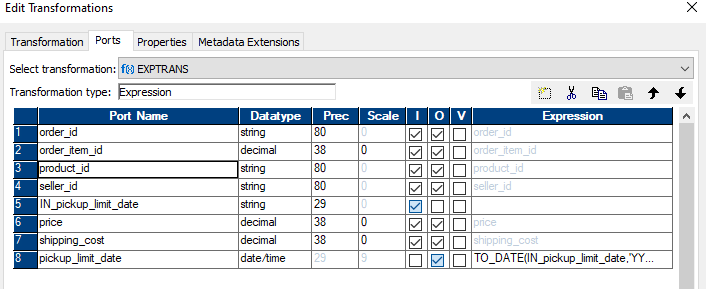
**DDL**

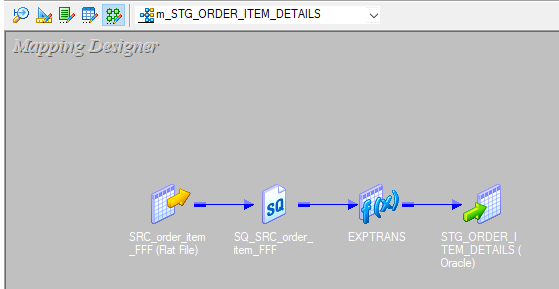
****

### 4. ETL Mapping

|  |
| --- |
|  |
|  |
|  |

**Transformations Applied:**

* **Expression Transformation:**



## PAYMENT TABLE

### 1. Mapping Overview(PAYMENT)

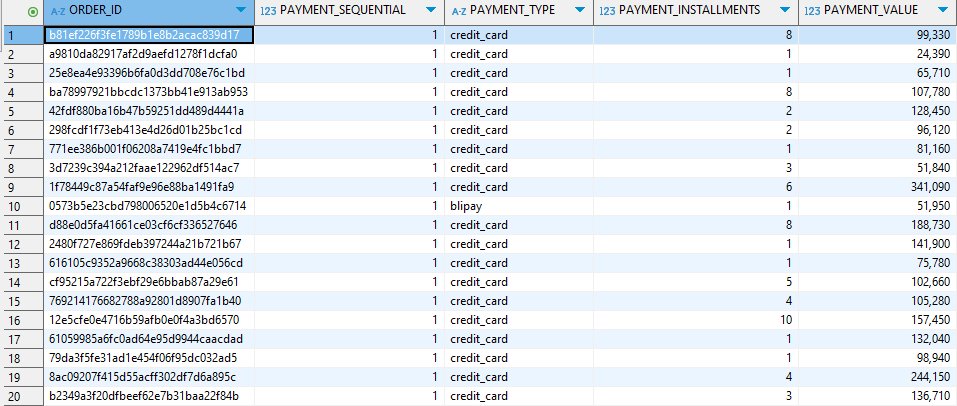
* **Mapping Name:** [m\_STG\_PAYMENT]
* **Source File:** [SRC\_payment\_FFF]
* **Target Table:** [STG\_PAYMENT]
* **Description:**[DATA IS LOADED FROM SOURCE TO STAGING TABLE]

### 2. Source File Details

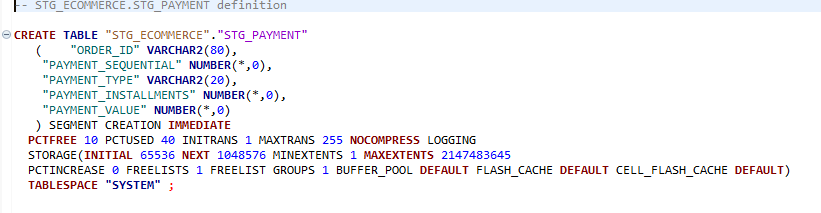
* **File Name:** [SRC\_payment\_FFF]
* **File Type:** [CSV]
* **Delimiter:** [Comma]
* **Column Structure**

### 3. Target (Staging Table) Details

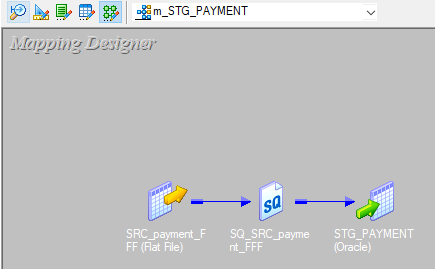
* **Table Name:** [STG\_PAYMENT]

**Column Structure**

**DDL**

****

### 4. ETL Mapping

****

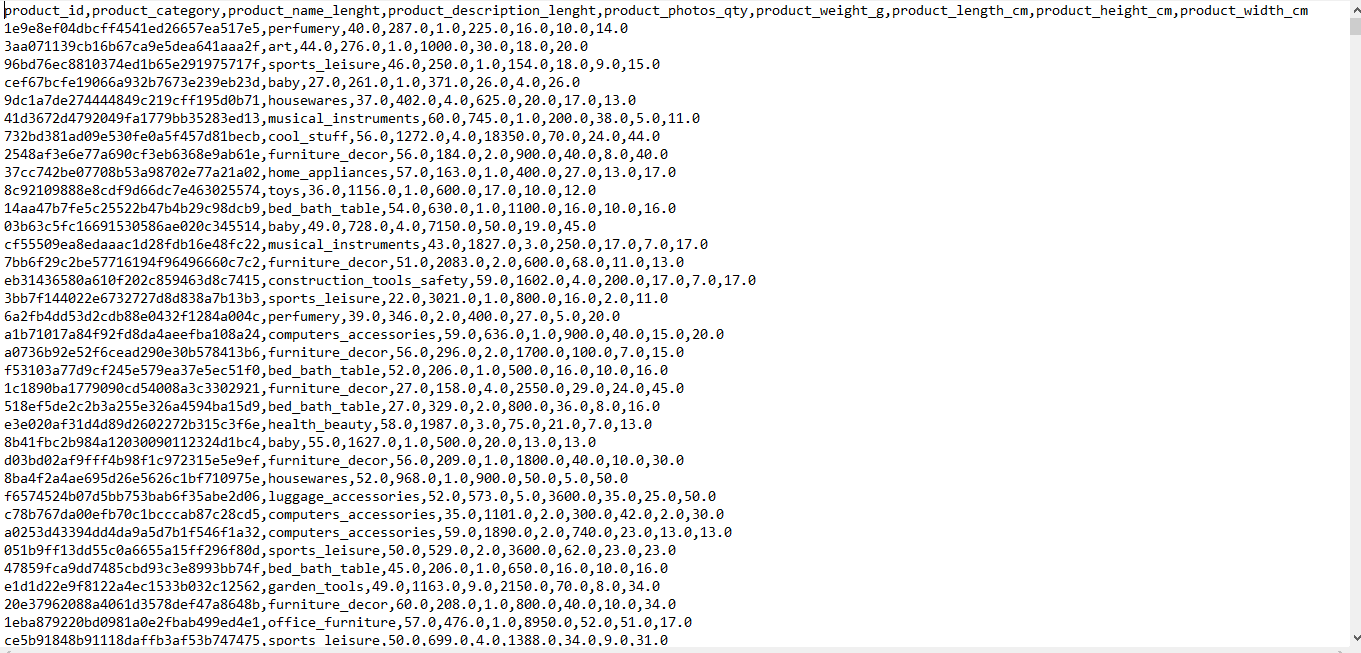
## PRODUCTS TABLE

### 1. Mapping Overview(PRODUCTS)

* **Mapping Name:** [m\_STG\_PRODUCTS]
* **Source File:** [SRC\_products\_FF]
* **Target Table:** [STG\_PRODUCTS]
* **Description:**[ DATA IS LOADED FROM SOURCE TO STAGING TABLE]

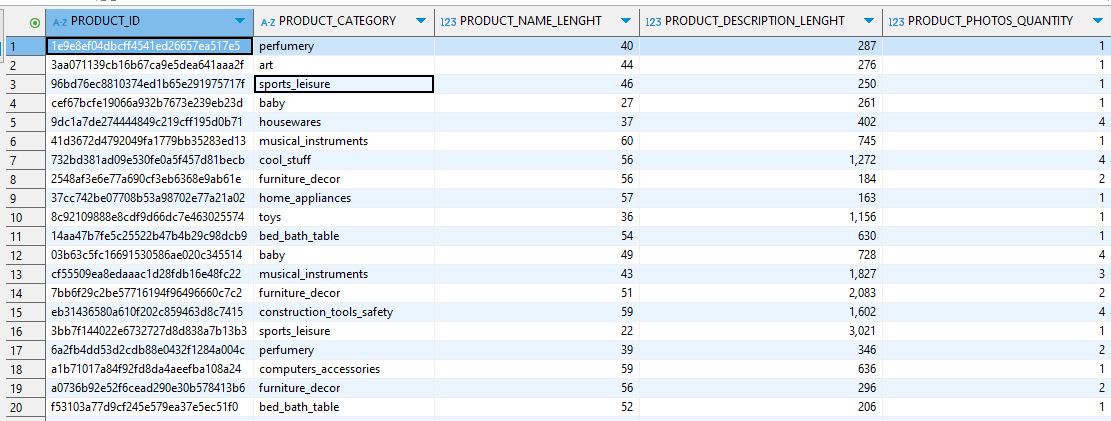
### 2. Source File Details

* **File Name:** [SRC\_products\_FF]
* **File Type:** [CSV]
* **Delimiter:** [Comma]

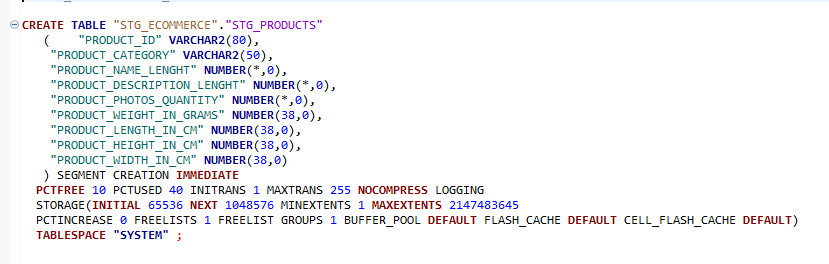
**Column Structure**

### 3. Target (Staging Table) Details

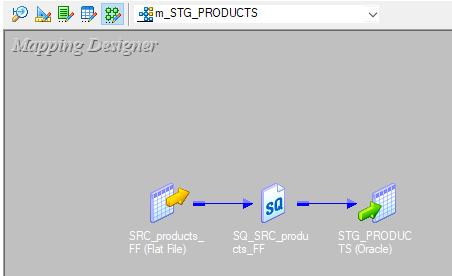
* **Table Name:** [STG\_PRODUCTS]

**Column Structure**

**DDL**

****

### 4. ETL Mapping



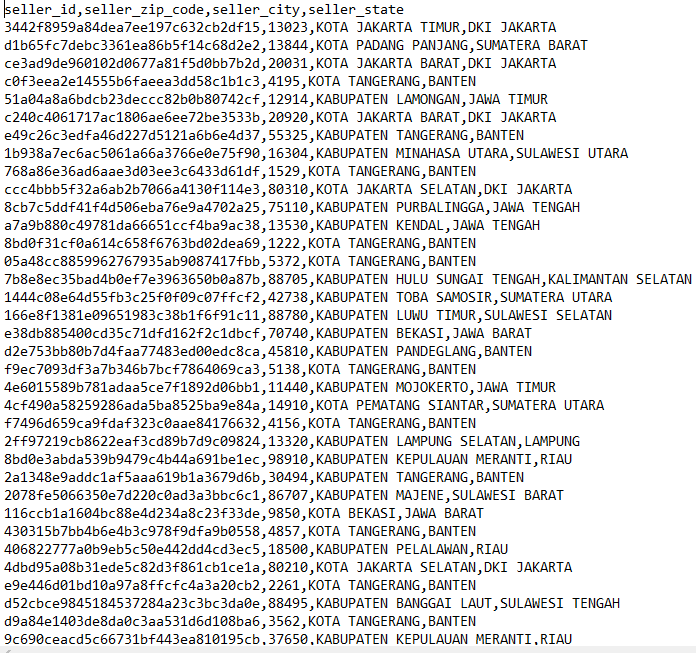
## SELLER TABLE

### 1. Mapping Overview(SELLER)

* **Mapping Name:** [m\_STG\_SELLER\_DETAILS]
* **Source File:** [SRC\_seller\_FF]
* **Target Table:** [STG\_SELLER\_DETAILS]
* **Description:**[ DATA IS LOADED FROM SOURCE TO STAGING TABLE]

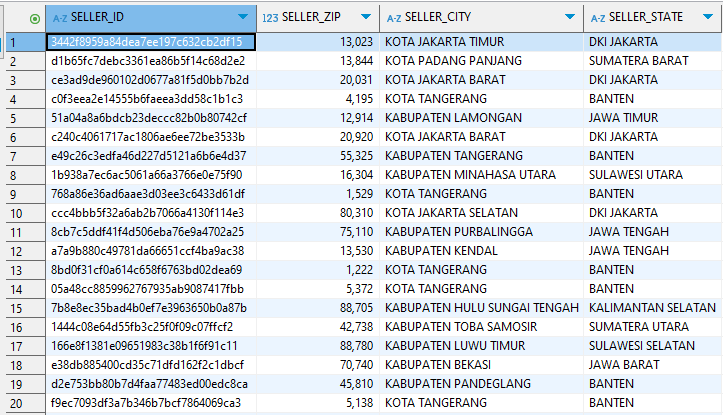
### 2. Source File Details

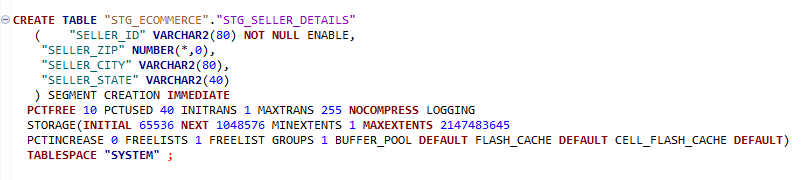
* **File Name:** [SRC\_seller\_FF]
* **File Type:** [CSV]
* **Delimiter:** [Comma]

**Column Structure**

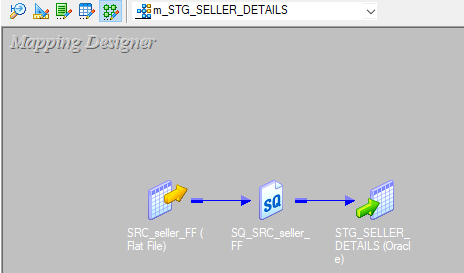
### 3. Target (Staging Table) Details

* **Table Name:** [STG\_SELLER\_DETAILS]

**Column Structure**

**DDL**

### 4. ETL Mapping



## USER TABLE

### 1. Mapping Overview(USER)

* **Mapping Name:** [m\_STG\_USER\_DETAILS]
* **Source File:** [SRC\_user\_FF]
* **Target Table:** [STG\_USER\_DETAILS]
* **Description:**[ DATA IS LOADED FROM SOURCE TO STAGING TABLE]

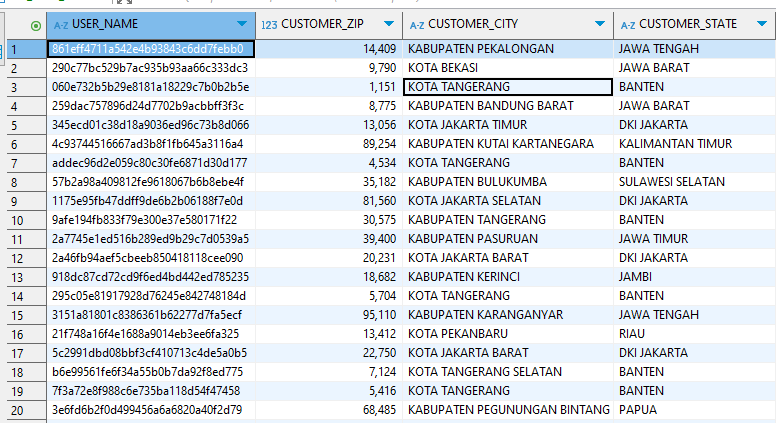
### 2. Source File Details

* **File Name:** [SRC\_user\_FF]
* **File Type:** [CSV]
* **Delimiter:** [Comma]

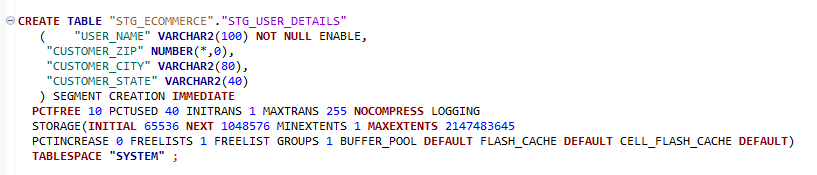
**Column Structure**

### 3. Target (Staging Table) Details

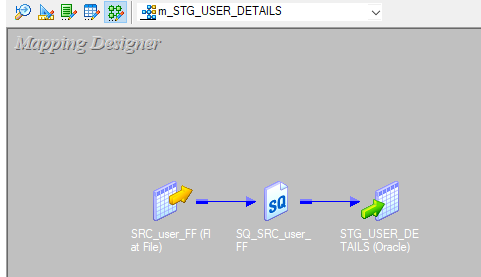
* **Table Name:** [STG\_USER\_DETAILS]

**Column Structure**

**DDL**

****

### 4. ETL Mapping

****

# Introduction to the Data Warehouse Layer

A **Data Warehouse (DWH)** is a centralized repository that stores integrated data from multiple sources, primarily for reporting, analysis, and decision-making. Unlike transactional databases, which are optimized for CRUD operations (Create, Read, Update, Delete), a Data Warehouse is optimized for analytical queries, historical data storage, and business intelligence (BI) reporting.

## Key Characteristics of the Data Warehouse Layer

1. **S ubject-Oriented**:
   * Organizes data by business subject areas (e.g., Sales, Orders, Customers).
2. **Integrated**:
   * Combines data from various operational systems (OLTP, external files, APIs).
3. **Non-Volatile**:
   * Once data is loaded, it is not updated or deleted frequently.
4. **Time-Variant**:
   * Maintains historical data to support trend analysis and forecasting.

## Data Warehouse Architecture in ETL

The **ETL (Extract, Transform, Load) process** consists of:

1. **Staging Layer**:
   * Temporary storage for raw, unprocessed data.
2. **Data Warehouse Layer**:
   * Centralized storage of **cleansed, transformed, and structured** data.
3. **Data Marts**:
   * Subsets of the warehouse optimized for specific business units (e.g., Sales Mart).

## Data Warehouse Layer in My Project

**Objective**

The goal of my project is to **design and implement a Data Warehouse layer** that stores structured data for analytics and reporting. Data is extracted from operational sources, staged temporarily, transformed, and then loaded into **fact and dimension tables** in the Data Warehouse.

## Data Warehouse Components in My Project

This project follows a **star schema** structure:

* **One Fact Table (FCT\_ORDER)** → Captures order lifecycle events.
* **Six Dimension Tables:**
  + DIM\_USER → Stores customer details.
  + DIM\_SELLER → Tracks seller information.
  + DIM\_PRODUCT → Stores product details.
  + DIM\_PAYMENT → Tracks payment methods and installment details.
  + DIM\_FEEDBACK → Stores customer feedback on orders.
  + DIM\_ORDER\_ITEM → Captures details of items within each order.
* **Supporting Time & Date Dimensions:**
  + DIM\_DATE → Enables date-based analysis.
  + DIM\_TIME → Tracks time-based events.

## ETL Process for My Data Warehouse

* **Extraction**: Data is pulled from staging tables like STG\_ORDER\_DETAILS.
* **Transformation**: Business rules are applied (e.g., key generation, lookups, data type conversion).
* **Loading**: Transformed data is inserted into warehouse tables like FCT\_ORDER and DIM\_USER.

## Why Data Integrity is Critical in My Project?

Since only **ETL Informatica** interacts with the database (no direct user access), ensuring **data integrity** is key. To maintain quality:

* **Primary and foreign key constraints** are applied at the ETL level.
* **Validation rules** prevent incorrect/missing data from entering the warehouse.
* **Error handling** logs rejected records for further analysis.

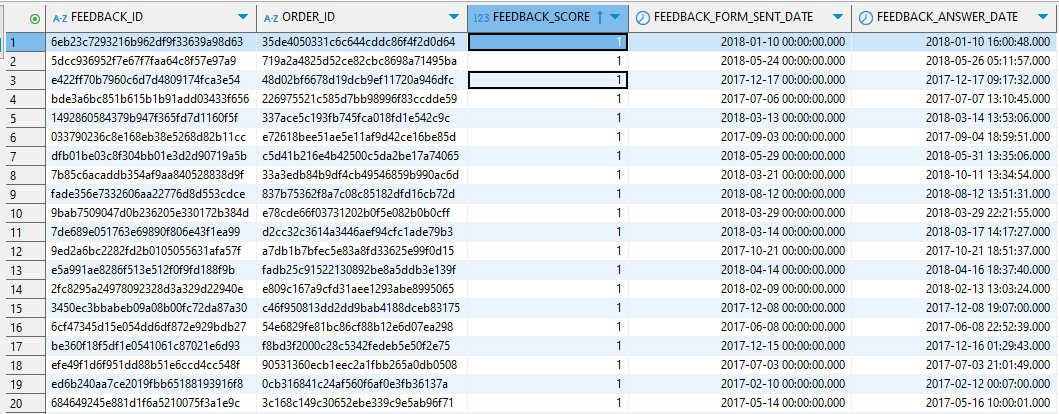
# ETL Mapping Documentation (Data Warehouse Layer)

## FEEDBACK TABLE

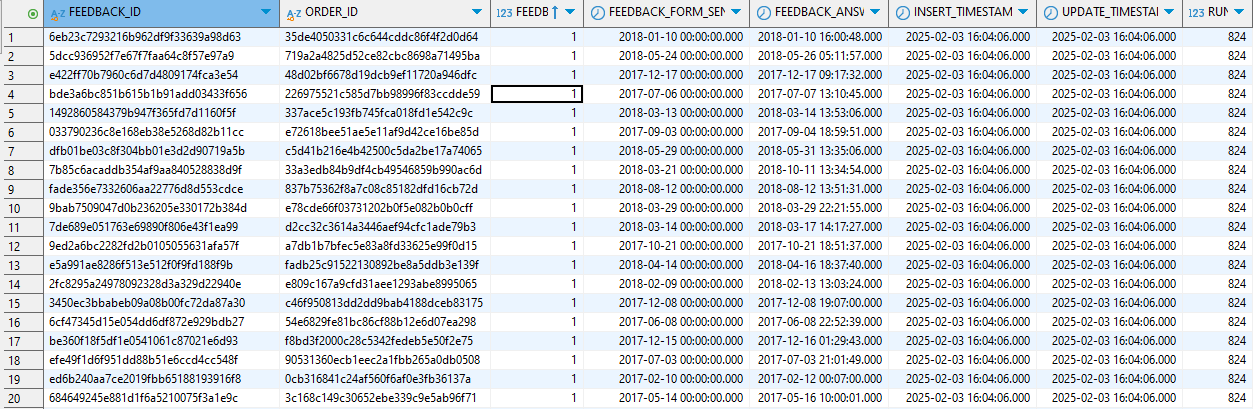
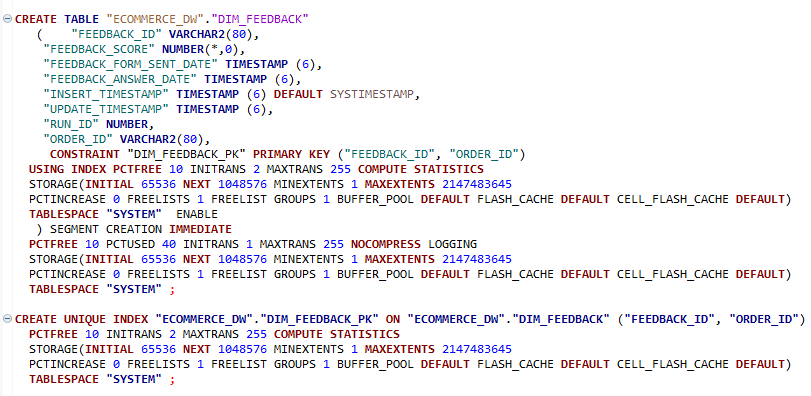
### 1. Mapping Overview(DIM\_FEEDBACK)

* **Mapping Name**: [m\_DIM\_FEEDBACK\_SCD\_T1]
* **Source Table**: [STG\_FEEDBACK]
* **Target Table**: [DIM\_FEEDBACK]
* **Description**: [ ]

### 2. Source (Staging Table) Details

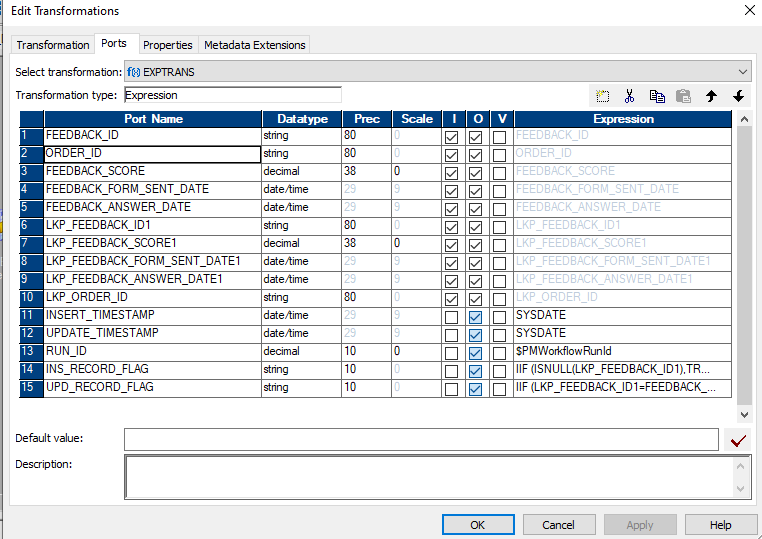
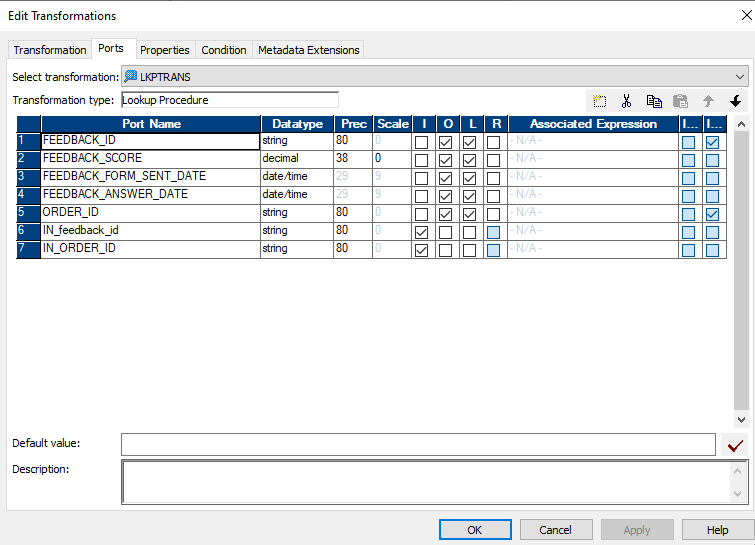
* **Table Name**: [STG\_FEEDBACK]
* **Column Structure**:

### 3. Target (Data Warehouse Table) Details

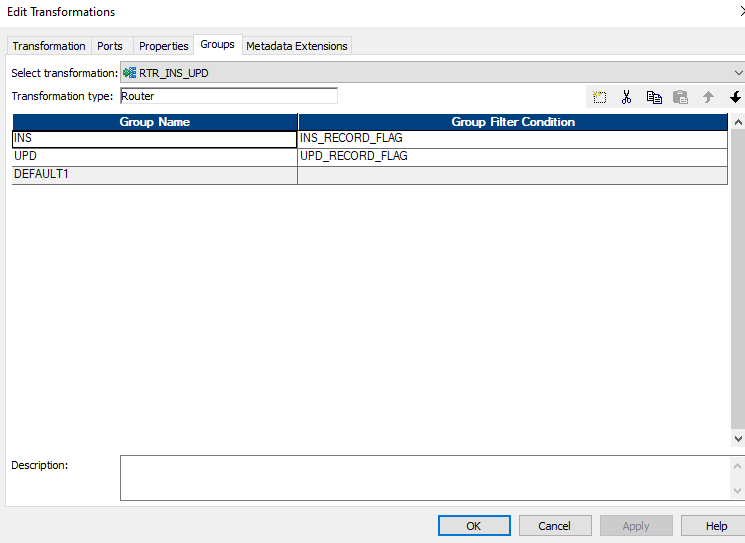
* **Table Name**: [DIM\_FEEDBACK]
* **Table Type**: [Dimension]
* **Column Structure**:
* **DDL **
* **Primary Key**: [FEEDBACK\_ID,ODER\_ID)
* **Foreign Keys** : [ORDER\_ID IN FACT TABLE REFERENCES ORDER\_ID IN DIM\_FEEDBACK]
* **Indexes**: [ON PRIMARY KEY]

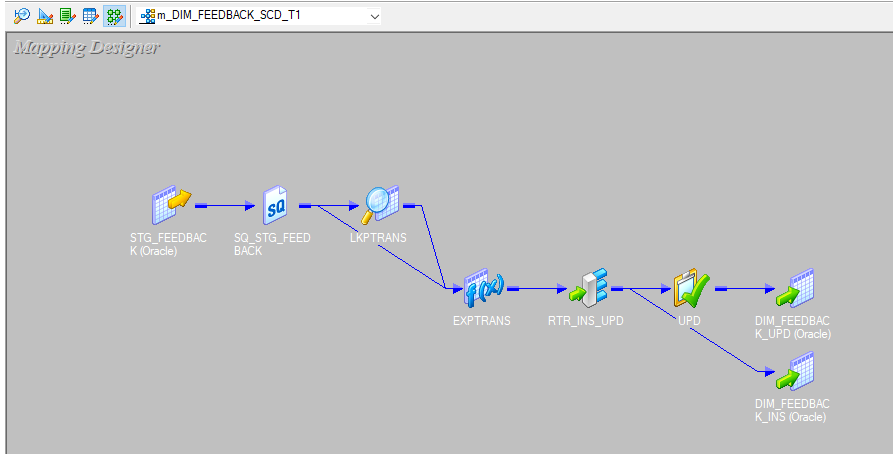
### 4. ETL Mapping Logic

**Transformations Applied**

* **Expression Transformation**
* **Lookup Transformation**
* **Handling Slowly Changing Dimension (SCD) Type 1 in Your Data Warehouse**
* In **SCD Type 1**, when there is a change in an existing record, we **update the existing record** instead of keeping history. If the record does not exist, we **insert it as a new record**.

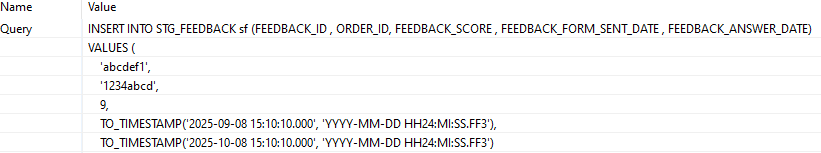
Lookup Transformation is **essential** in handling Slowly Changing Dimensions (SCD) because it helps check if a record **already exists** in the target dimension table (DIM\_TABLE). This is necessary to determine whether we should UPDATE OR INSERT:

* **ROUTER Transformation**:
* **Router Transformation** is an **active and connected** transformation in Informatica that allows you to route data into multiple output groups based on specified conditions
* WE Use Router Transformation to Separate Updates and Inserts

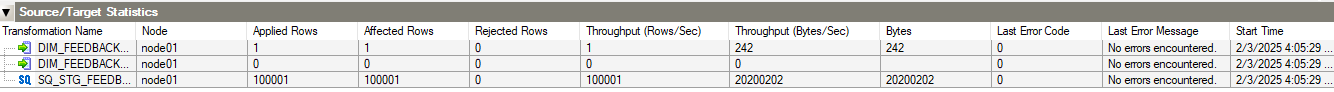
**MAPPING**:

### SCD Type 1 Verification Process(DIM\_FEEDBACK)

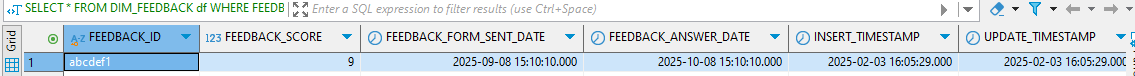
#### Step 1: Insert a New Record into STG\_FEEDBACK



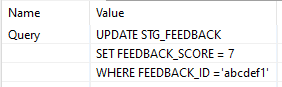
#### Step 2: Run the Informatica Workflow



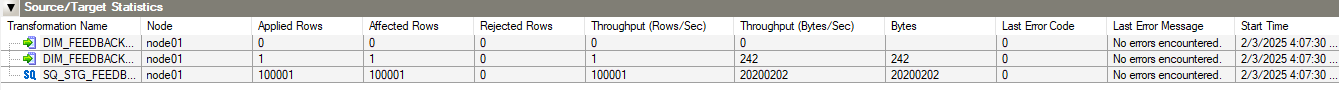
#### Step 3: Verify Data in DIM\_FEEDBACK Table



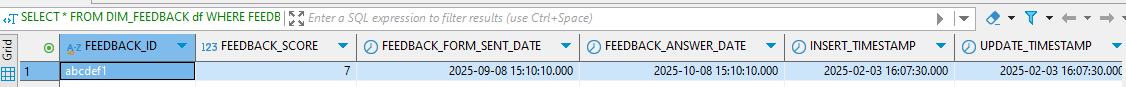
#### Step 4: Update the Record in STG\_FEEDBACK



#### Step 5: Re-run the Informatica Workflow



#### Step 6: Verify the Update in DIM\_FEEDBACK



#### Summary of SCD Type 1 Behavior

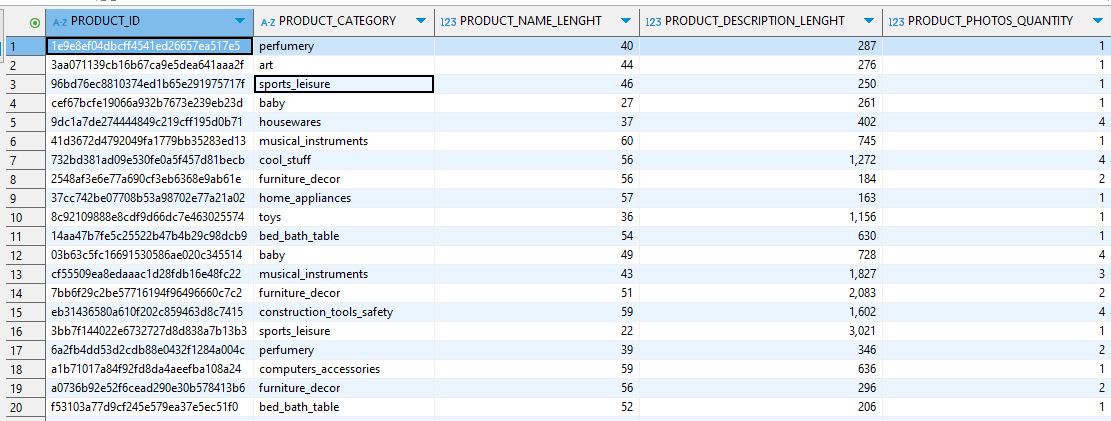
|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
| Insert | Insert a new record in STG\_FEEDBACK | The new record appears in DIM\_FEEDBACK |
| Run Workflow | Execute Informatica mapping | Data moves from staging to warehouse |
| Verify Insert | Check DIM\_FEEDBACK | Record exists with FEEDBACK\_SCORE = 9 |
| Update | Modify the FEEDBACK\_SCORE | Data in STG\_FEEDBACK changes |
| Run Workflow Again | Re-run Informatica mapping | Changes propagate to DIM\_FEEDBACK |
| Verify Update | Check DIM\_FEEDBACK | Old FEEDBACK\_SCORE = 9 is replaced by 7 |

## PRODUCTS TABLE

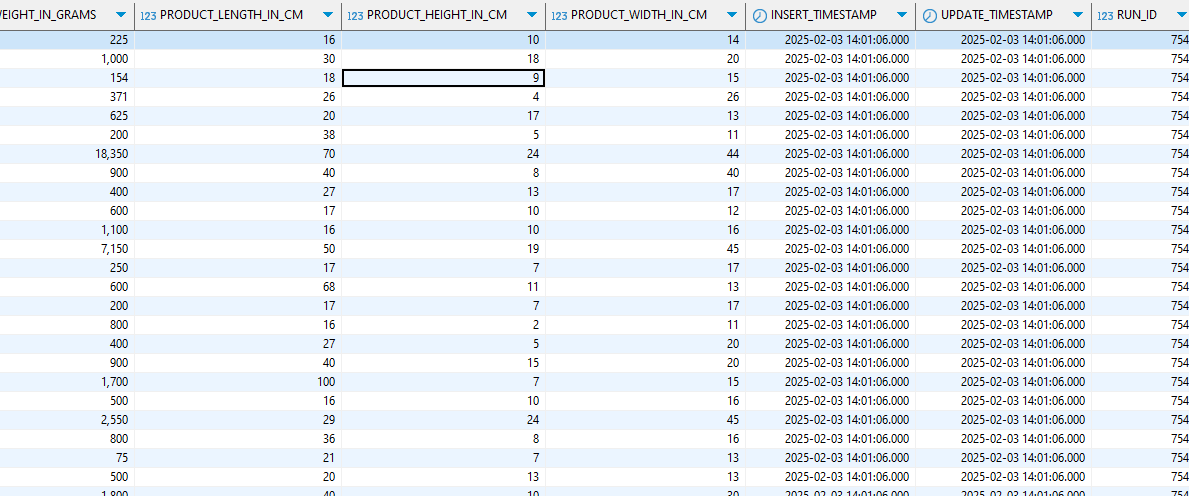
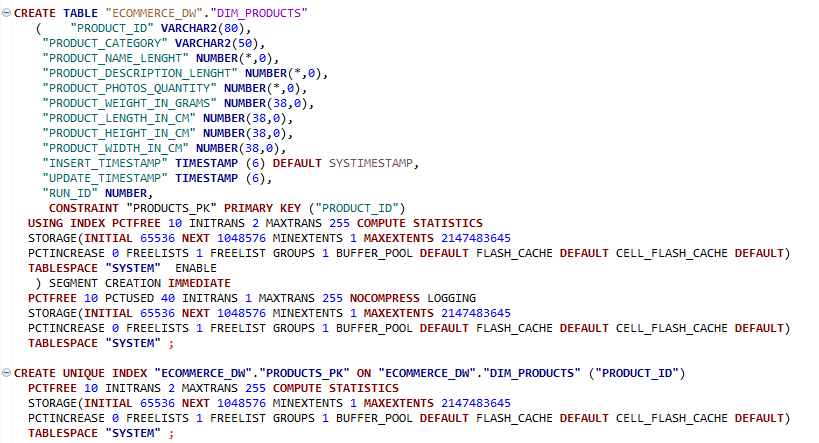
### 1. Mapping Overview(DIM\_PRODUCTS)

* **Mapping Name**: [m\_DIM\_PRODUCTS\_SCD\_T1]
* **Source Table**: [STG\_PRODUCTS]
* **Target Table**: [DIM\_PRODUCTS]
* **Description**: [ ]

### 2. Source (Staging Table) Details

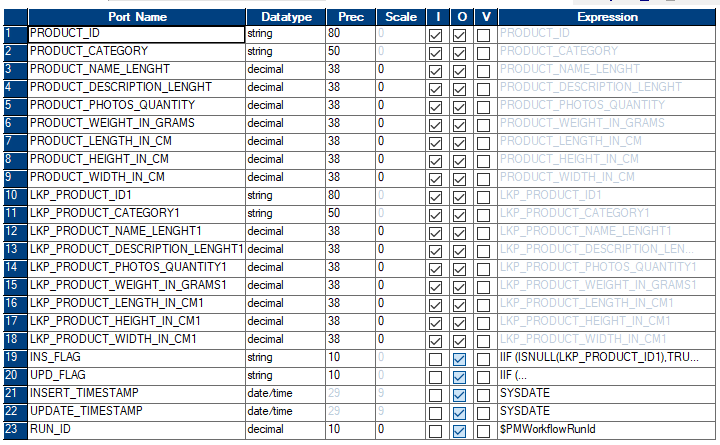
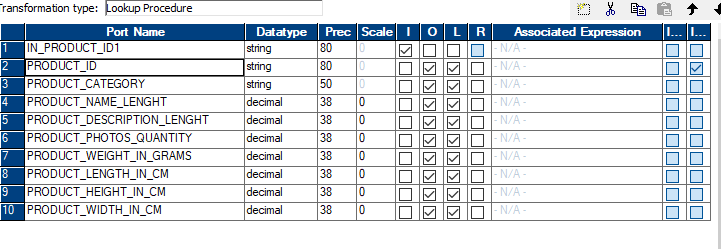
* **Table Name**: [STG\_PRODUCTS]
* **Column Structure**:

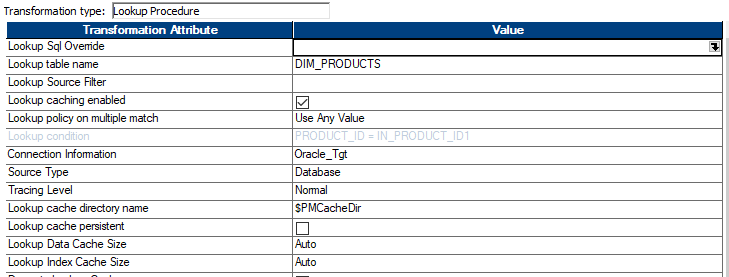
### 3. Target (Data Warehouse Table) Details

* **Table Name**: [DIM\_PRODUCTS]
* **Table Type**: [Dimension]
* **Column Structure**:
* **DDL**
* **Primary Key**: [PRODUCT\_ID)
* **Foreign Keys** : []
* **Indexes**: [ON PRIMARY KEY]

### 4. ETL Mapping Logic

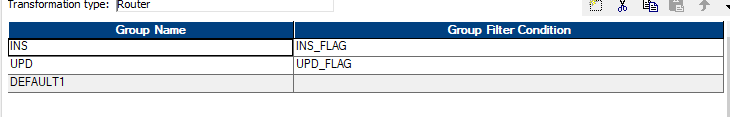
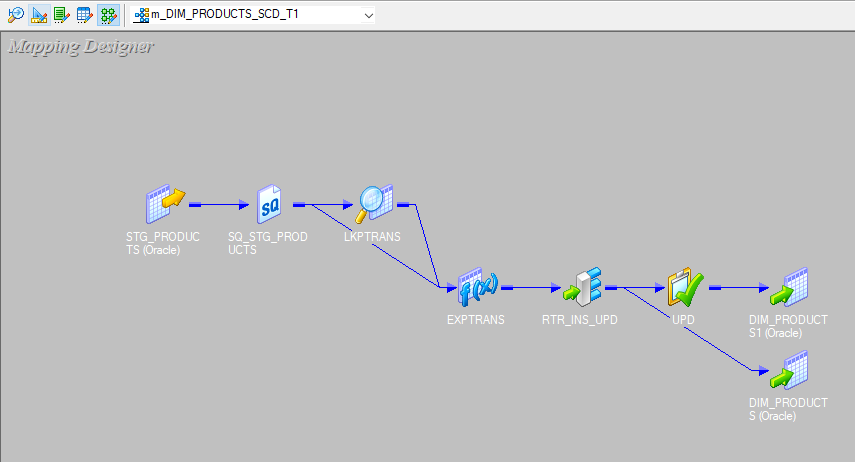
**Transformations Applied**

* **Expression Transformation**
* **Lookup Transformation**



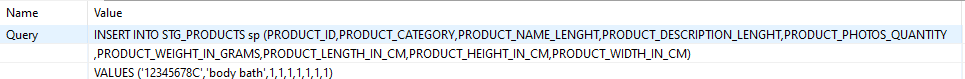
* **Handling Slowly Changing Dimension (SCD) Type 1 in Your Data Warehouse**
* In **SCD Type 1**, when there is a change in an existing record, we **update the existing record** instead of keeping history. If the record does not exist, we **insert it as a new record**.

Lookup Transformation is **essential** in handling Slowly Changing Dimensions (SCD) because it helps check if a record **already exists** in the target dimension table (DIM\_TABLE). This is necessary to determine whether we should UPDATE OR INSERT:

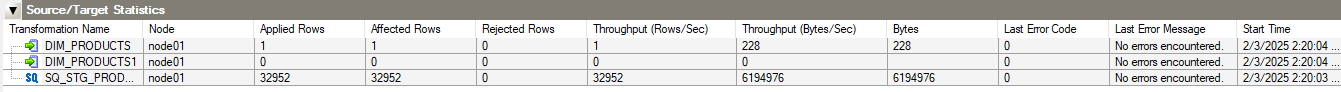
* **ROUTER Transformation**:
* **Router Transformation** is an **active and connected** transformation in Informatica that allows you to route data into multiple output groups based on specified conditions
* WE Use Router Transformation to Separate Updates and Inserts
* **MAPPING**:

### SCD Type 1 Verification Process (DIM\_PRODUCTS)

#### Step 1: Insert a New Record into STG\_PRODUCTS



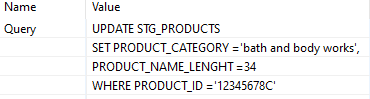
#### Step 2: Run the Informatica Workflow



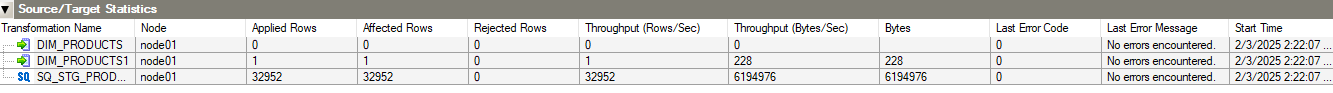
#### Step 3: Verify Data in DIM\_PRODUCTS Table



#### Step 4: Update the Record in STG\_PRODUCTS



#### Step 5: Re-run the Informatica Workflow



#### Step 6: Verify the Update in DIM\_PRODUCTS



#### Summary of SCD Type 1 Behavior

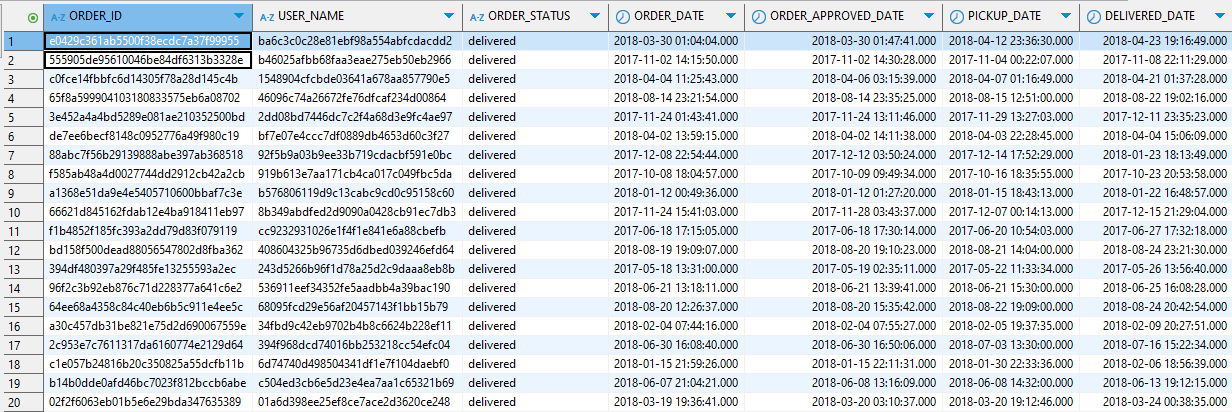
|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
| Insert | Insert a new record in STG\_PRODUCTS | The new record appears in DIM\_PRODUCTS |
| Run Workflow | Execute Informatica mapping | Data moves from staging to warehouse |
| Verify Insert | Check DIM\_PRODUCTS | Record exists with PRODUCTS\_CATEGORY = body |
| Update | Modify the PRODUCTS\_CATEGORY | Data in STG\_PRODUCTS changes |
| Run Workflow Again | Re-run Informatica mapping | Changes propagate to DIM\_PRODUCTS |
| Verify Update | Check DIM\_PRODUCTS | Old PRODUCTS\_CATEGORY = body bath is replaced by bath and body works |

## ORDER TABLE

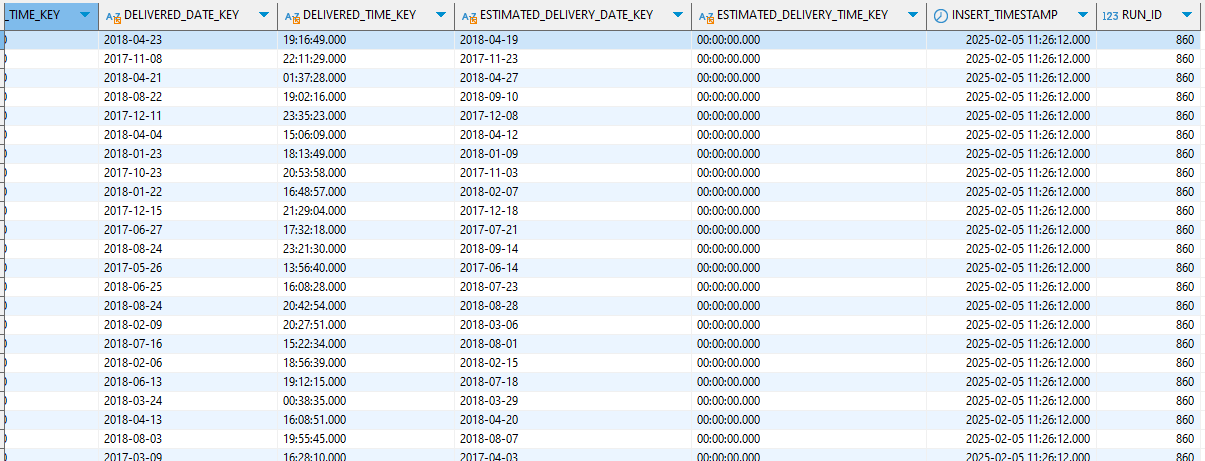
### 1. Mapping Overview(FCT\_ORDER)

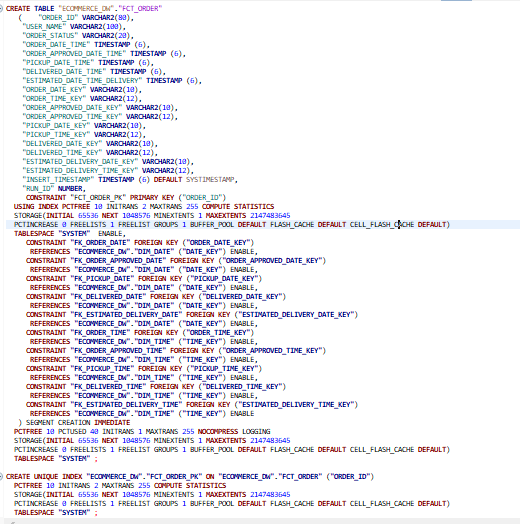
* **Mapping Name**: [m\_FCT\_ORDER\_SCD\_T1]
* **Source Table**: [STG\_ORDER\_DETAILS]
* **Target Table**: [FCT\_ORDER]
* **Description**: [ ]

### 2. Source (Staging Table) Details

* **Table Name**: [STG\_ORDER\_DETAILS]
* **Column Structure**:

### 3. Target (Data Warehouse Table) Details

* **Table Name**: [FCT\_ORDER]
* **Table Type**: [FACT]
* **Column Structure**:
* **DDL**

****

* **Primary Key**: [ORDER\_ID)
* **Foreign Keys** : [

ORDER\_ID REFERENCES DIM\_FEEDBACK(ORDER\_ID)

ORDER\_ID REFERENCES DIM\_PAYMENT(ORDER\_ID)

ORDER\_ID REFERENCES DIM\_ORDER\_ITEM(ORDER\_ID)

USER\_NAME REFERENCES DIM\_USER(USER\_NAME)

ORDER\_DATE\_KEY REFERENCES DIM\_DATE( DATE\_KEY)

ORDER\_APPROVED\_DATE\_KEY REFERENCES DIM\_DATE( DATE\_KEY)]

PICKUP\_DATE\_KEY REFERENCES DIM\_DATE( DATE\_KEY)]

DELIVERED\_DATE\_KEY REFERENCES DIM\_DATE( DATE\_KEY)]

ESTIMATED\_DELIEVERY\_DATE\_KEY REFERENCES DIM\_DATE( DATE\_KEY)

ORDER\_TIME\_KEY REFERENCES DIM\_TIME( TIME\_KEY )

ORDER\_APPROVED\_TIME\_KEY REFERENCES DIM\_TIME( TIME\_KEY )

PICKUP\_TIME\_KEY REFERENCES DIM\_TIME( TIME\_KEY )

DELIVERED\_TIME\_KEY REFERENCES DIM\_TIME( TIME\_KEY )

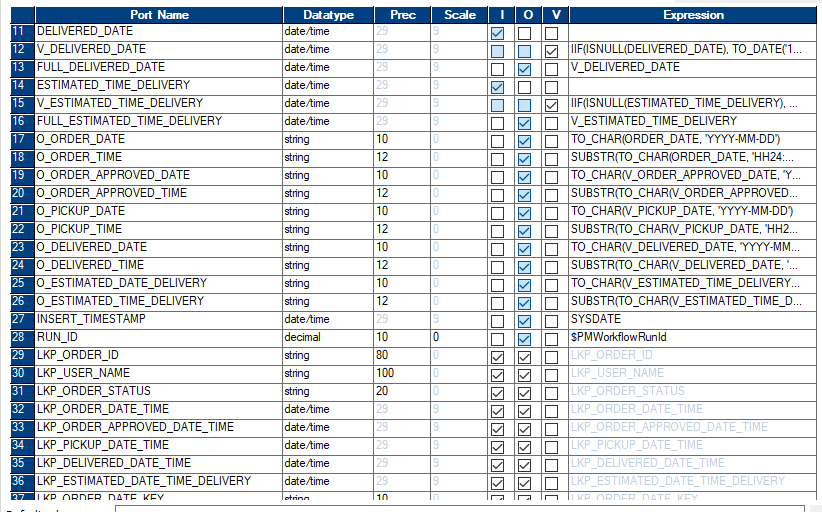
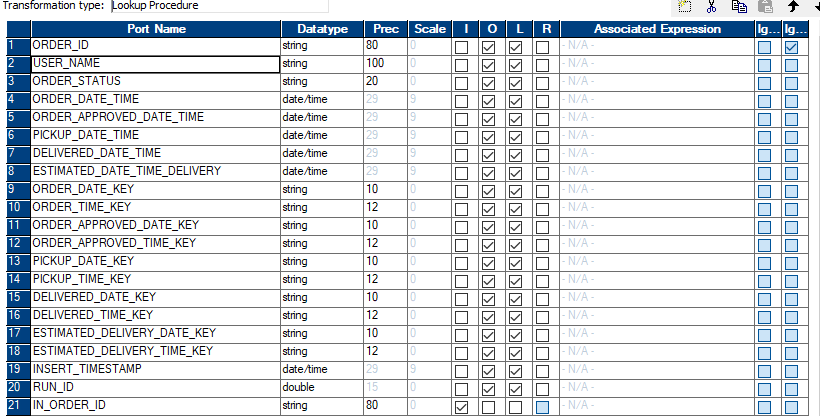
ESTIMATED\_DELIVERY\_TIME\_KEY REFERENCES DIM\_TIME( TIME\_KEY )

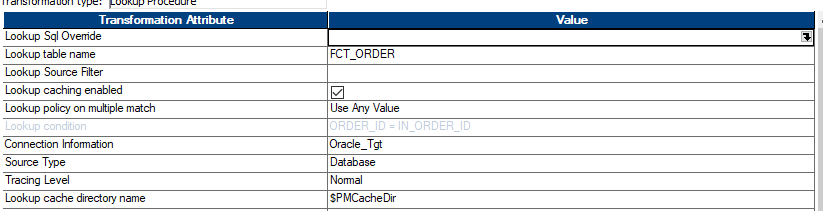
]

* **Indexes**: [ON PRIMARY KEY]

### 4. ETL Mapping Logic

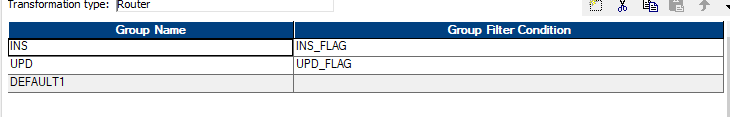
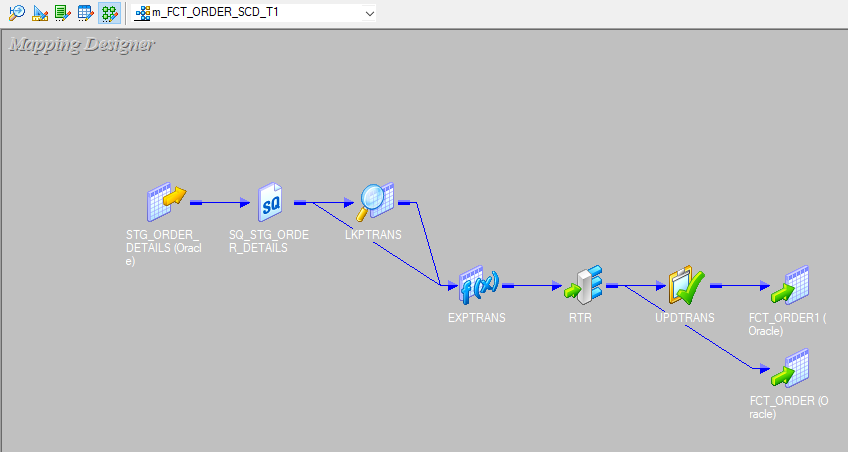
**Transformations Applied**

* **Expression Transformation**
* **Lookup Transformation**



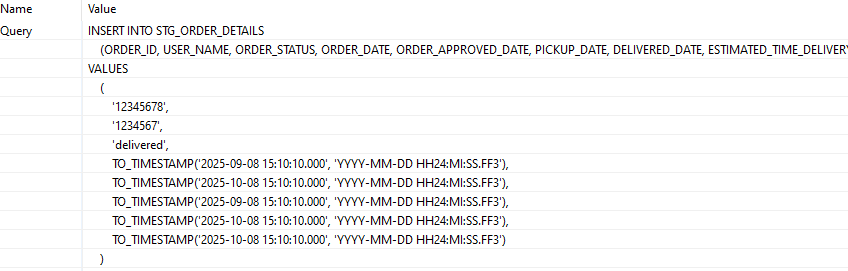
* **Handling Slowly Changing Dimension (SCD) Type 1 in Your Data Warehouse**
* In **SCD Type 1**, when there is a change in an existing record, we **update the existing record** instead of keeping history. If the record does not exist, we **insert it as a new record**.

Lookup Transformation is **essential** in handling Slowly Changing Dimensions (SCD) because it helps check if a record **already exists** in the target FACT table (FACT\_TABLE). This is necessary to determine whether we should UPDATE OR INSERT:

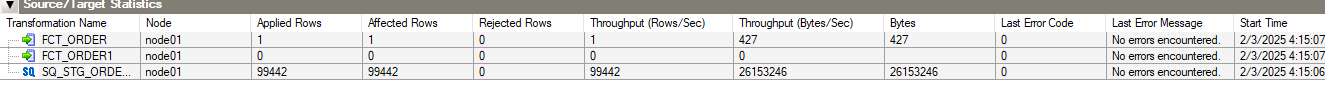
* **ROUTER Transformation**:
* **Router Transformation** is an **active and connected** transformation in Informatica that allows you to route data into multiple output groups based on specified conditions
* WE Use Router Transformation to Separate Updates and Inserts
* **MAPPING**: 

### SCD Type 1 Verification Process(FCT\_ORDER)

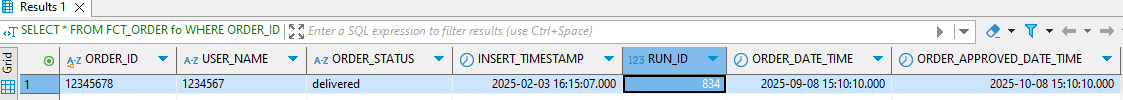
#### Step 1: Insert a New Record into STG\_ ORDER\_DETAILS



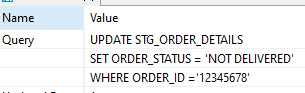
#### Step 2: Run the Informatica Workflow



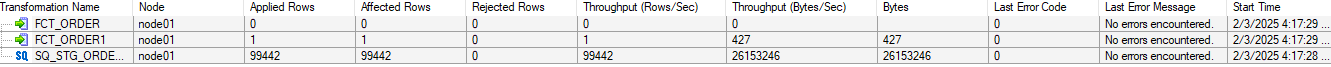
#### Step 3: Verify Data in FCT\_ORDER



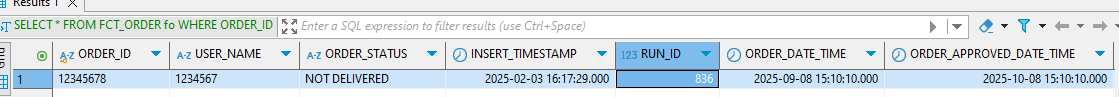
#### Step 4: Update the Record in STG\_ ORDER\_DETAILS



#### Step 5: Re-run the Informatica Workflow



#### Step 6: Verify the Update in FCT\_ORDER



#### Summary of SCD Type 1 Behavior

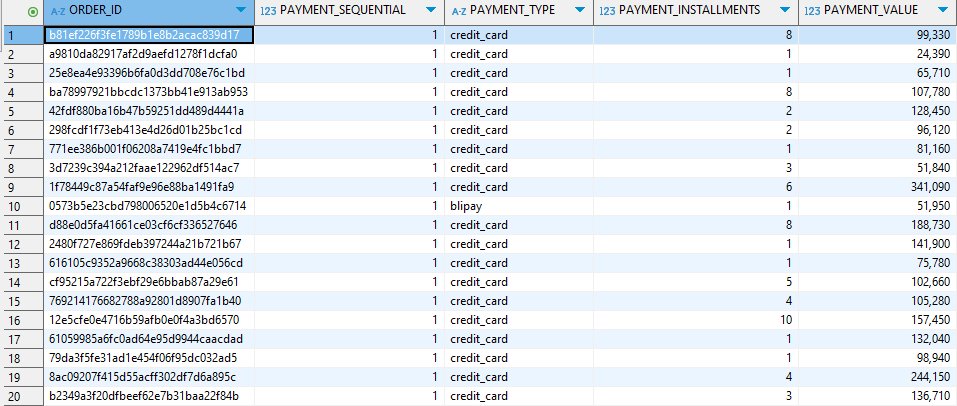
|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
| Insert | Insert a new record in STG\_ ORDER\_DETAILS | The new record appears in FCT\_ORDER |
| Run Workflow | Execute Informatica mapping | Data moves from staging to warehouse |
| Verify Insert | Check FCT\_ORDER | Record exists with ORDER\_STATUS = delivered |
| Update | Modify the ORDER\_STATUS | Data in STG\_ ORDER\_DETAILS changes |
| Run Workflow Again | Re-run Informatica mapping | Changes propagate to FCT\_ORDER |
| Verify Update | Check FCT\_ORDER | Old ORDER\_STATUS = delivered is replaced by not delivered |

## PAYMENT TABLE

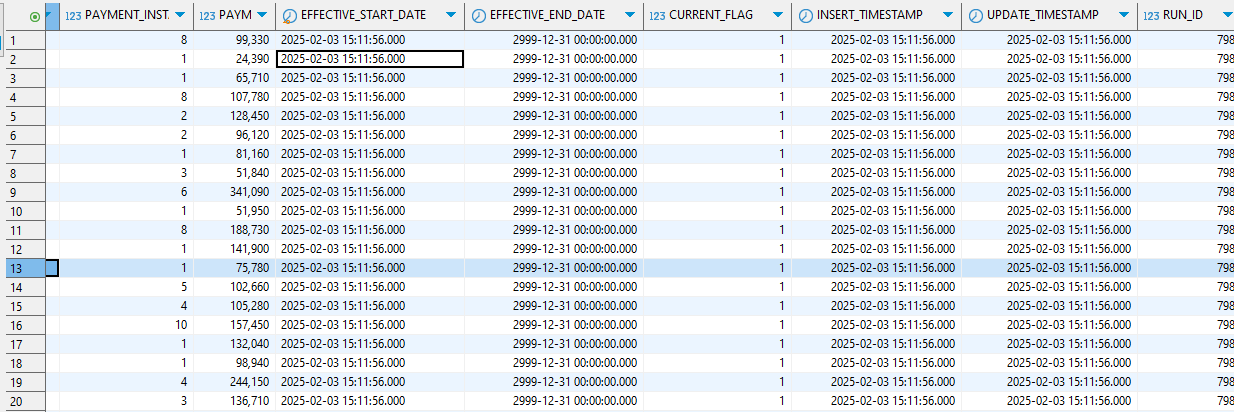
### 1. Mapping Overview(DIM\_PAYMENT)

* **Mapping Name**: [m\_DIM\_PAYMENT\_SCD\_T2]
* **Source Table**: [STG\_PAYMENT]
* **Target Table**: [DIM\_PAYMENT]
* **Description**:[ ]

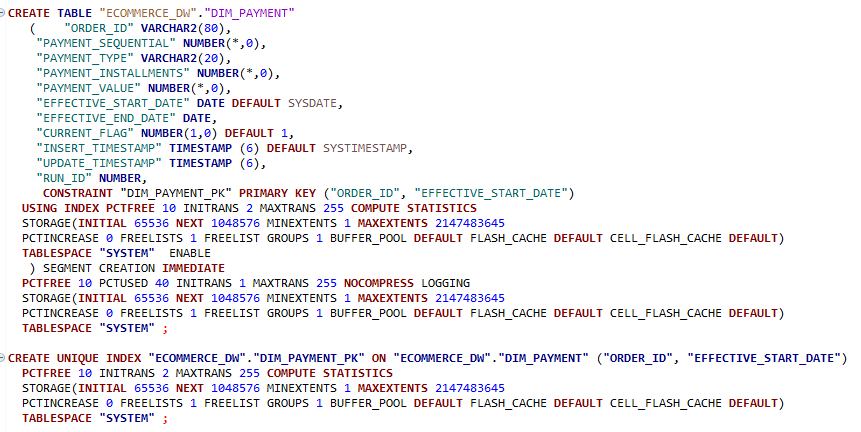
### 2. Source (Staging Table) Details

* **Table Name**: [STG\_PAYMENT]
* **Column Structure**:

### 3. Target (Data Warehouse Table) Details

* **Table Name**: [DIM\_PAYMENT]
* **Table Type**: [Dimension Table]
* **Column Structure**:

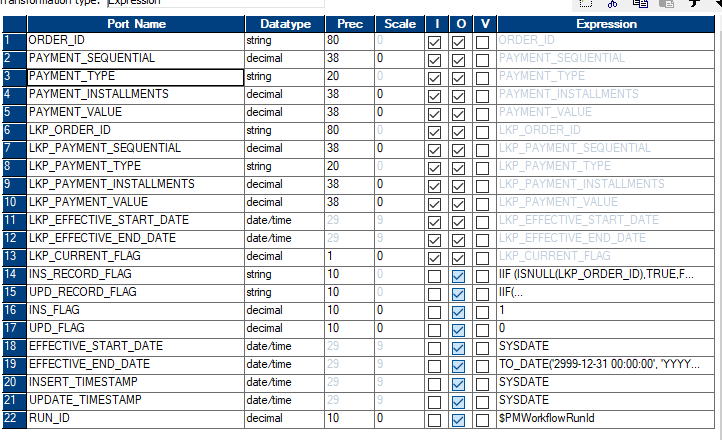
**DDL**

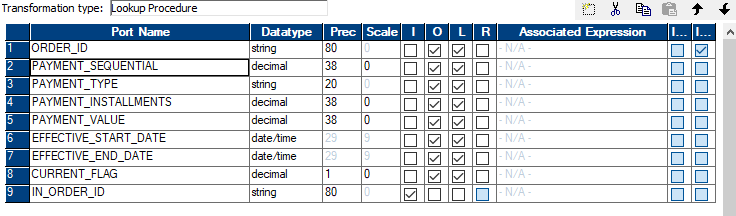
****

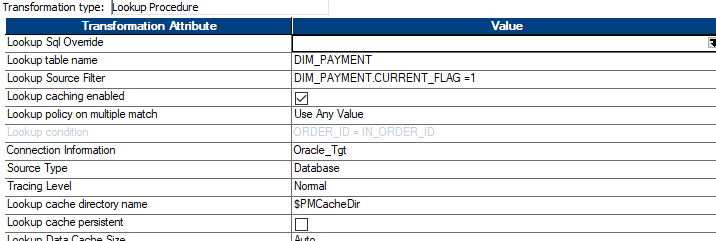
* **Primary Key**: [ORDER\_ID ,EFFECTIVE START DATE]
* **Foreign Keys** [ORDER\_ID IN FACT TABLE REFERENCES ORDER\_ID IN DIM\_PAYMENT]
* **Indexes**: [ON PRIMARY KEY]

### 4. ETL Mapping Logic

**Transformations Applied**

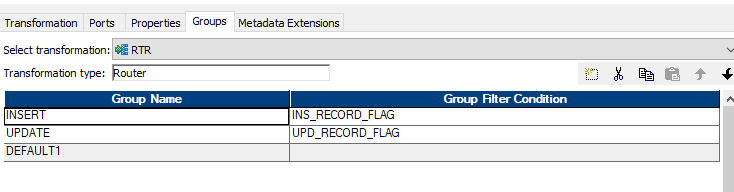
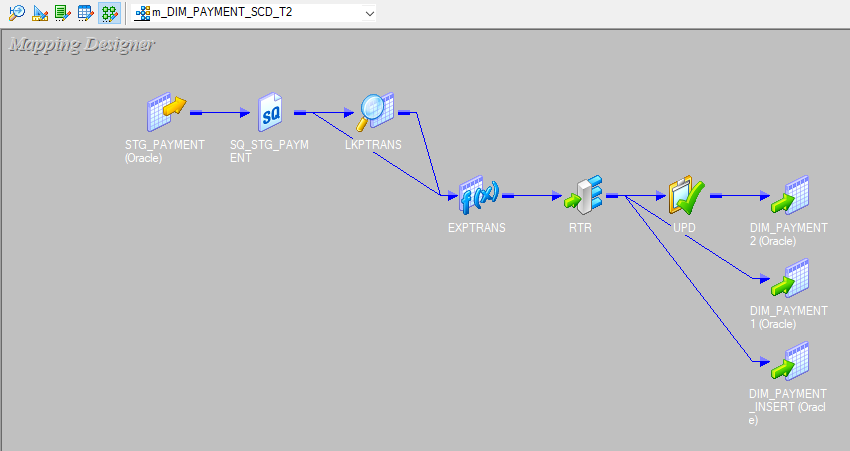
* **Expression Transformation**: 
* **Lookup Transformation**;

****



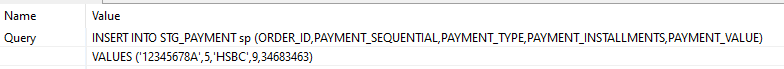
* In **SCD Type 2**, when a change occurs in an existing record, we **insert a new record** instead of updating the existing one.
* This allows us to **track historical changes** by maintaining multiple versions of the same record

Lookup Transformation is **essential** in handling Slowly Changing Dimensions (SCD) because it helps check if a record **already exists** in the target dimension table (DIM\_TABLE). This is necessary to determine whether we should UPDATE OR INSERT:

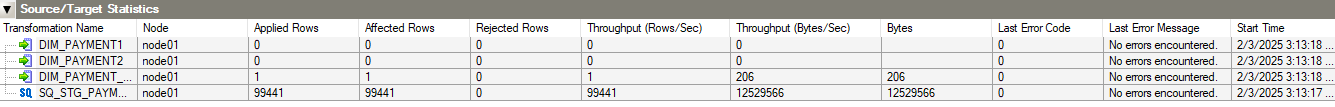
* **ROUTER Transformation**:
* **Router Transformation** is an **active and connected** transformation in Informatica that allows you to route data into multiple output groups based on specified conditions
* IN SUMMARY :WE Use Router Transformation to Separate Updates and Inserts
* **MAPPING**

### SCD Type 2 Verification Process(DIM\_PAYMENT)

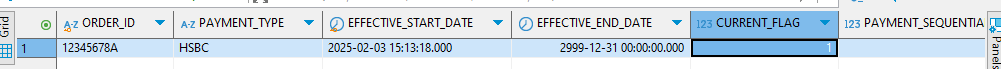
#### Step 1: Insert a New Record into STG\_PAYMENT



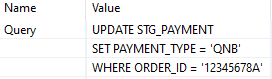
#### Step 2: Run the Informatica Workflow



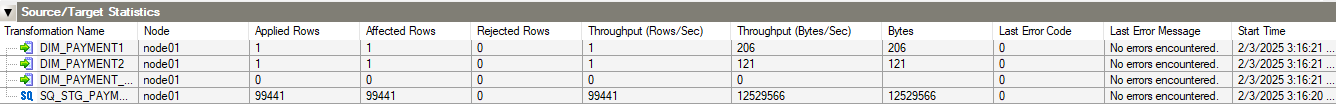
#### Step 3: Verify Data in DIM\_PAYMENT Table



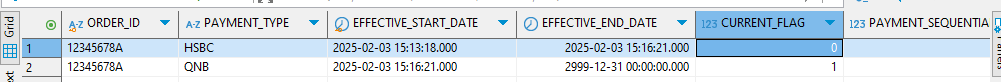
#### Step 4: Update the Record in STG\_PAYMENT



#### Step 5: Re-run the Informatica Workflow



#### Step 6: Verify the Update in DIM\_PAYMENT



#### Summary of SCD Type 2 Behavior

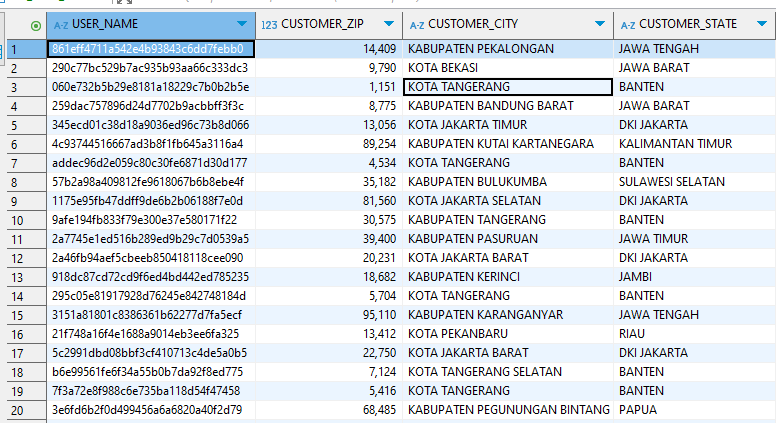
|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
| Insert | Insert a new record in STG\_PAYMENT | The new record appears in DIM\_PAYMENT with CURRENT\_FLAG = 1 |
| Run Workflow | Execute Informatica mapping | Data moves from staging to warehouse |
| Verify Insert | Check DIM\_PAYMENT | Record exists with `PAYMENT\_TYPE = 'HSBC'` |
| Update | Modify the `PAYMENT\_TYPE` | A new version is created in `DIM\_PAYMENT` |
| Run Workflow Again | Re-run Informatica mapping | A new row is inserted, and the old row is marked as historical |
| Verify Update | Check DIM\_PAYMENT | `PAYMENT\_TYPE = QNB in the new record, old record has CURRENT\_FLAG = 0 |

## USER TABLE

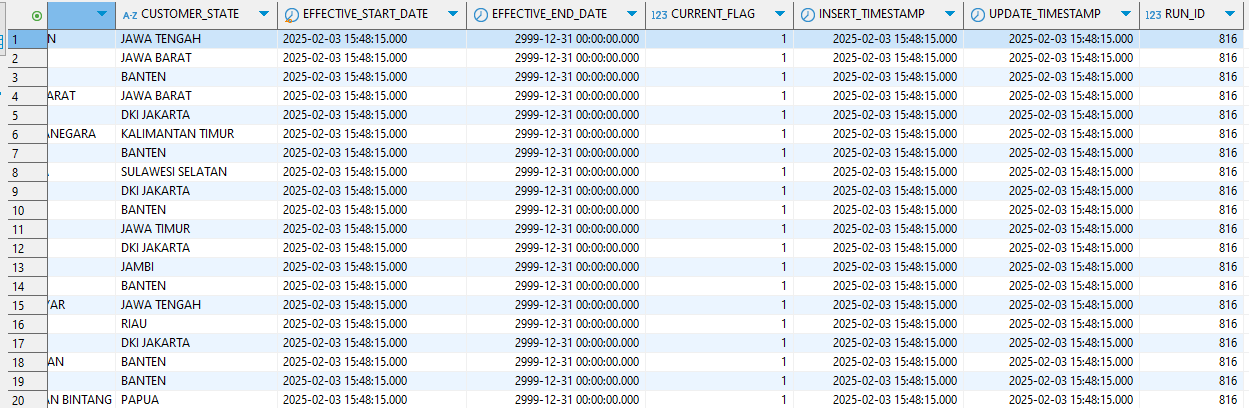
### 1. Mapping Overview(DIM\_USER)

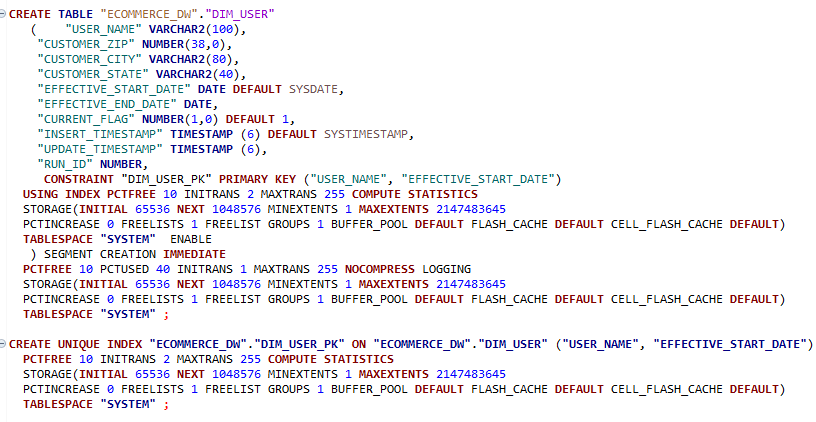
* **Mapping Name**: [m\_DIM\_USER\_SCD\_T2]
* **Source Table**: [STG\_USER\_DETAILS]
* **Target Table**: [DIM\_USER]
* **Description**: [ ]

### 2. Source (Staging Table) Details

* **Table Name**: [STG\_USER\_DETAILS]]
* **Column Structure**:

### 3. Target (Data Warehouse Table) Details

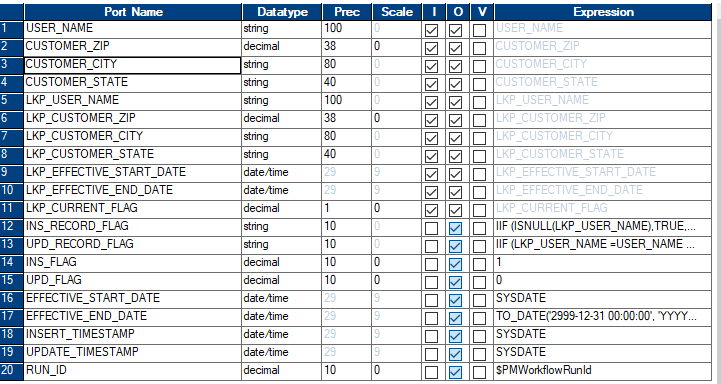
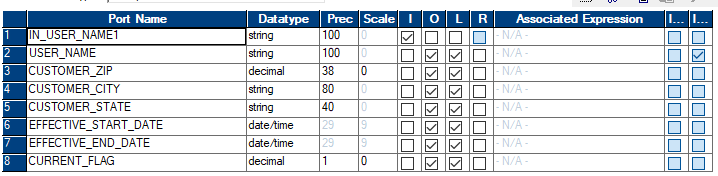
* **Table Name**: [DIM\_USER]
* **Table Type**: [Dimension]
* **Column Structure**:

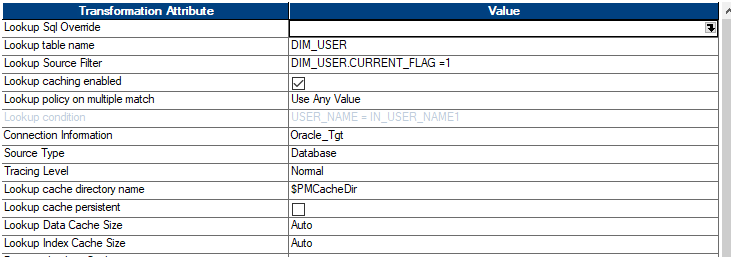
**DDL**

* **Primary Key**: [PRODUCT\_ID)
* **Foreign Keys** : [USER\_NAME , EFFECTIVE\_START\_DATE]
* **Indexes**: [ON PRIMARY KEY]

### 4. ETL Mapping Logic

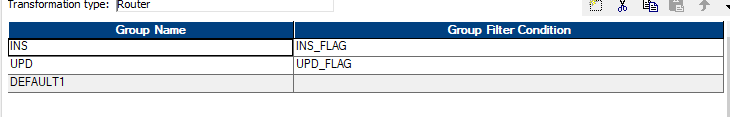
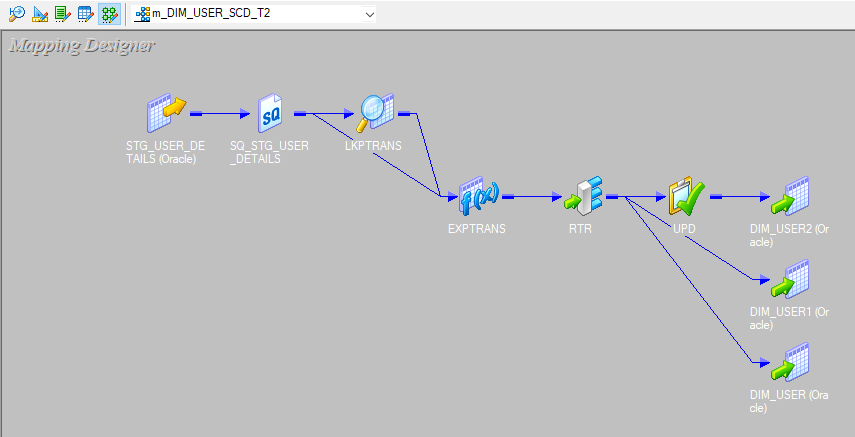
**Transformations Applied**

* **Expression Transformation**
* **Lookup Transformation**



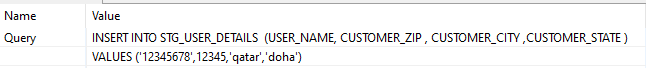
* In **SCD Type 2**, when a change occurs in an existing record, we **insert a new record** instead of updating the existing one.
* This allows us to **track historical changes** by maintaining multiple versions of the same record

Lookup Transformation is **essential** in handling Slowly Changing Dimensions (SCD) because it helps check if a record **already exists** in the target dimension table (DIM\_TABLE). This is necessary to determine whether we should UPDATE OR INSERT:

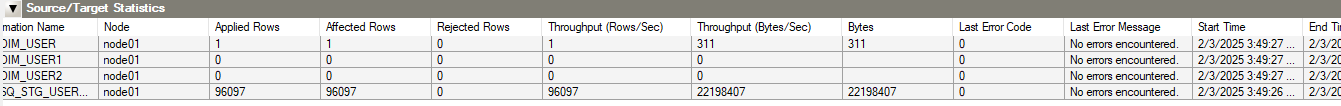
* **ROUTER Transformation**:
* **Router Transformation** is an **active and connected** transformation in Informatica that allows you to route data into multiple output groups based on specified conditions
* WE Use Router Transformation to Separate Updates and Inserts
* **MAPPING**:

### SCD Type 2 Verification Process (DIM\_USER)

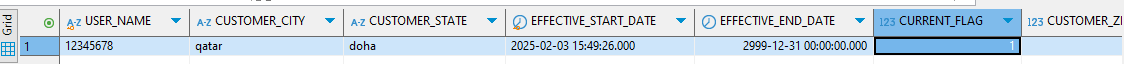
#### Step 1: Insert a New Record into **STG\_USER\_DETAILS**



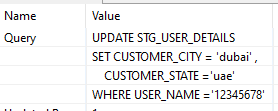
#### Step 2: Run the Informatica Workflow



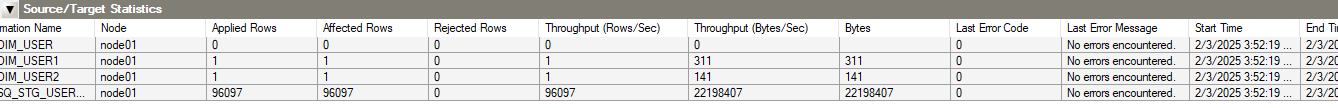
#### Step 3: Verify Data in **DIM\_USER** Table



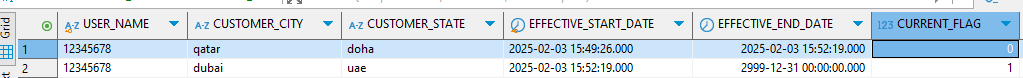
#### Step 4: Update the Record in **STG\_USER\_DETAILS**



#### Step 5: Re-run the Informatica Workflow



#### Step 6: Verify the Update in **DIM\_USER**



#### Summary of SCD Type 2 Behavior

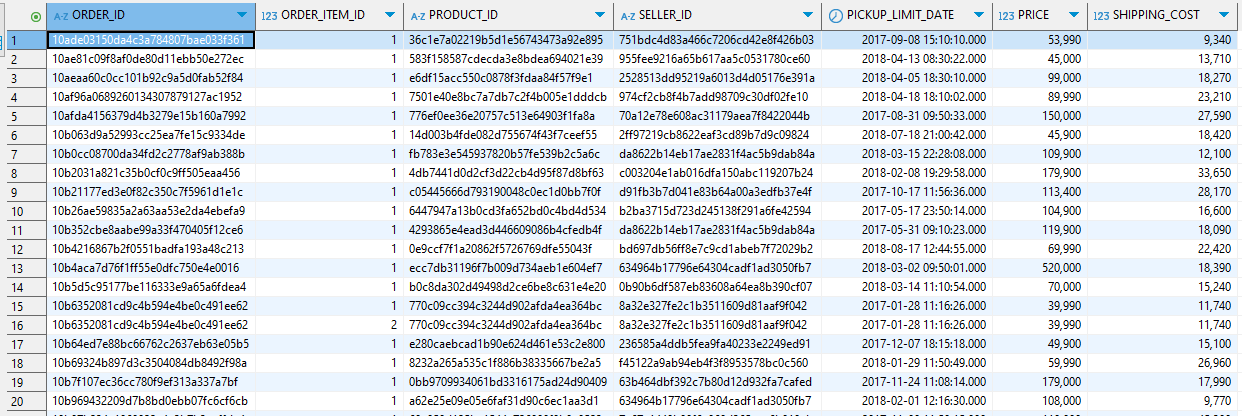
|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
| Insert | Insert a new record in  STG\_USER\_DETAILS | The new record appears in DIM\_USER with CURRENT\_FLAG =1 |
| Run Workflow | Execute Informatica mapping | Data moves from staging to warehouse |
| Verify Insert | Check DIM\_USER | Record exists where  CUSTOMER\_CITY = qatar  CUSTOMER\_STATE = doha |
| Update | Modify  CUSTOMER\_CITY  CUSTOMER\_STATE | A new version is created in `DIM\_USER |
| Run Workflow Again | Re-run Informatica mapping | A new row is inserted, and the old row is marked as historical |
| Verify Update | Check  DIM\_USER | NEW Record exists where  -CUSTOMER\_CITY = dubai  -CUSTOMER\_STATE = uae  -OLD RECORD FLAG = 0 |

## ORDER ITEM TABLE

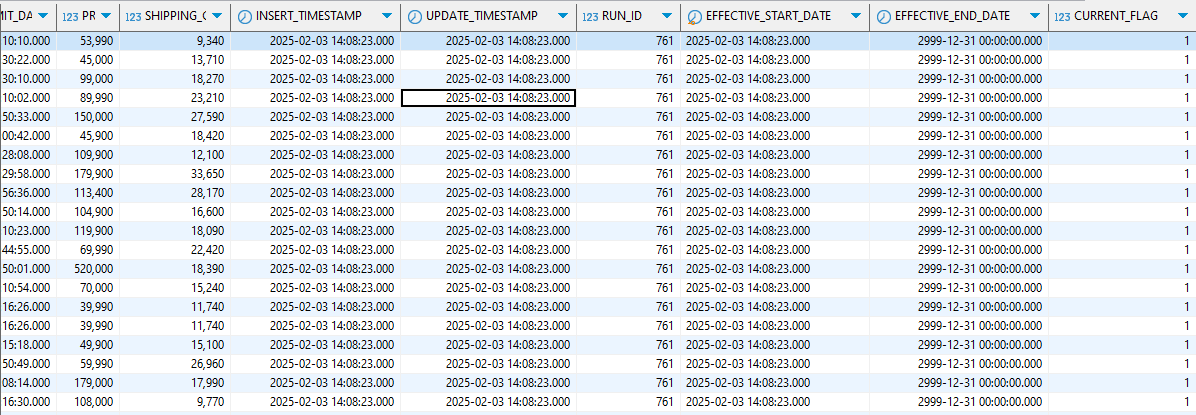
### 1. Mapping Overview(ORDER\_ITEM)

* **Mapping Name**: [m\_DIM\_ORDER\_ITEM\_SCD\_T2]
* **Source Table**: [STG\_ORDER\_ITEM\_DETAILS]
* **Target Table**: [DIM\_ORDER\_ITEM]
* **Description**:[ ]

### 2. Source (Staging Table) Details

* **Table Name**: [STG\_ORDER\_ITEM\_DETAILS]
* **Column Structure**:

### 3. Target (Data Warehouse Table) Details

* **Table Name**: [DIM\_ORDER\_ITEM]
* **Table Type**: [Dimension Table]
* **Column Structure**:

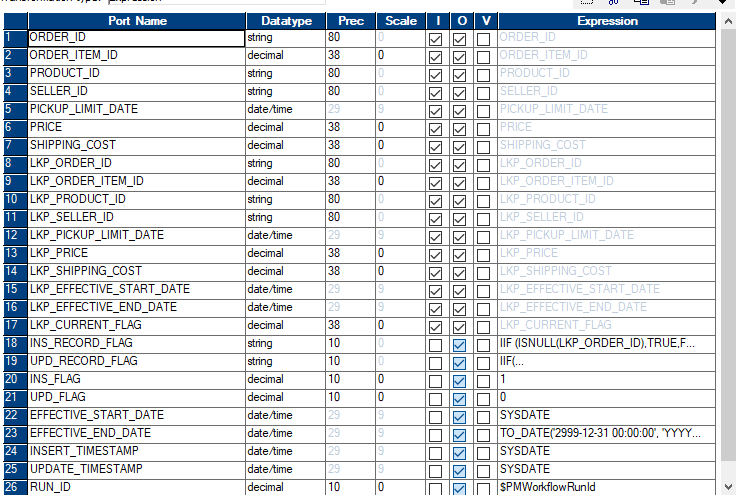
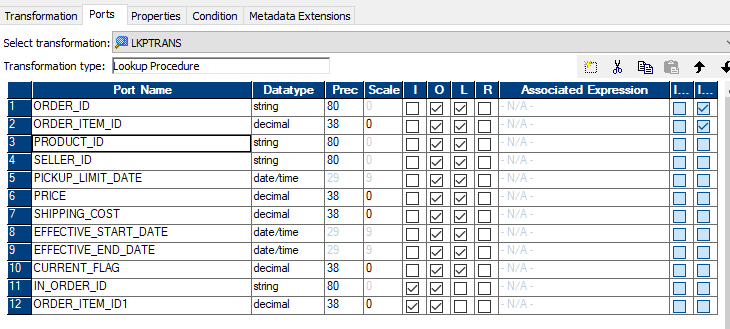
**DDL**

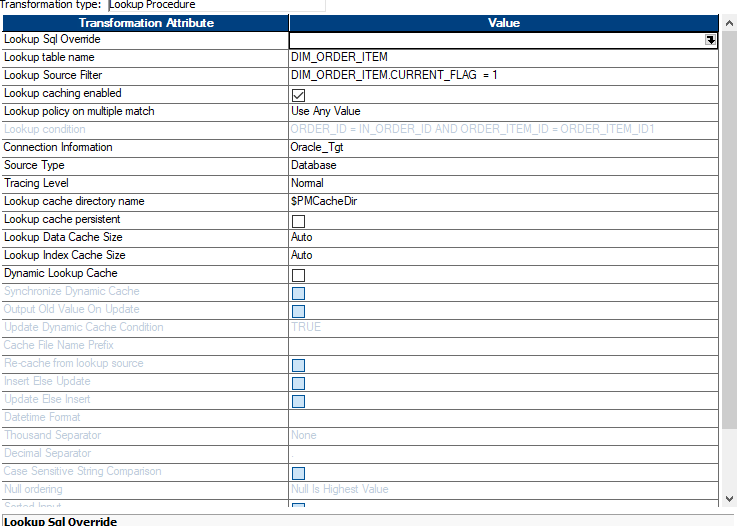
****

* **Primary Key**: [ORDER\_ID , ORDER\_ITEM\_ID,EFFECTIVE START DATE]
* **Foreign Keys** [PRODUCT\_ID, SELLER\_ID]
* **Indexes**: [ON PRIMARY KEY]

### 4. ETL Mapping Logic

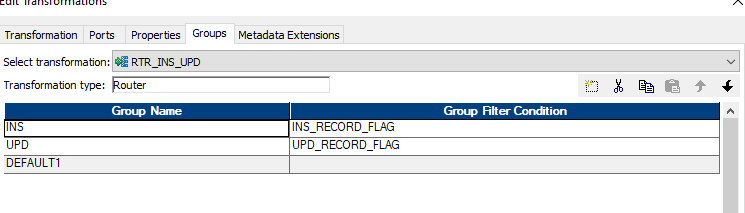
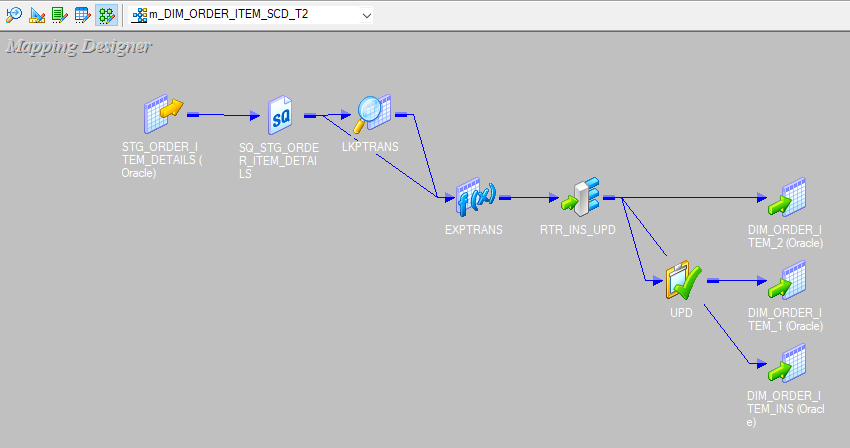
**Transformations Applied**

* **Expression Transformation**: 
* **Lookup Transformation**;

****

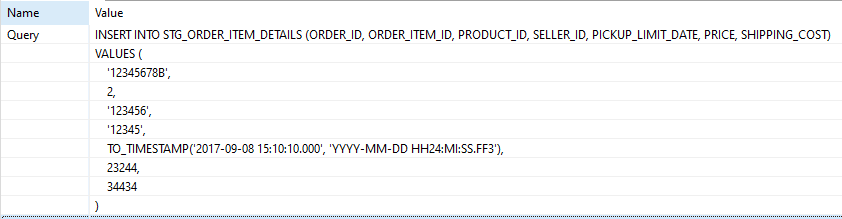
* In **SCD Type 2**, when a change occurs in an existing record, we **insert a new record** instead of updating the existing one.
* This allows us to **track historical changes** by maintaining multiple versions of the same record

Lookup Transformation is **essential** in handling Slowly Changing Dimensions (SCD) because it helps check if a record **already exists** in the target dimension table (DIM\_TABLE). This is necessary to determine whether we should UPDATE OR INSERT:

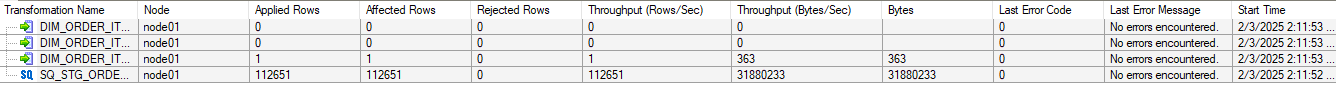
* **ROUTER Transformation**:
* **Router Transformation** is an **active and connected** transformation in Informatica that allows you to route data into multiple output groups based on specified conditions
* IN SUMMARY :WE Use Router Transformation to Separate Updates and Inserts
* **MAPPING**

### SCD Type 2 Verification Process (DIM\_ORDER\_ITEM)

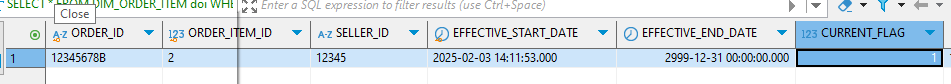
#### Step 1: Insert a New Record into **STG\_ORDER\_ITEM\_DETAILS**



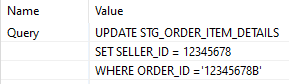
#### Step 2: Run the Informatica Workflow



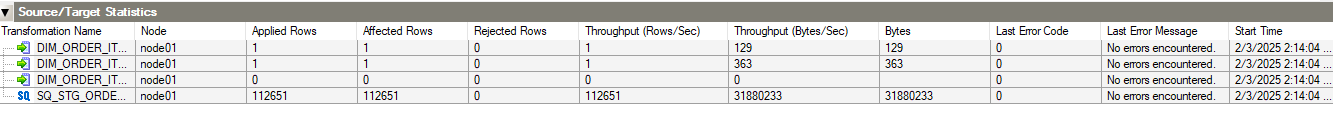
#### Step 3: Verify Data in **DIM\_ORDER\_ITEM** Table



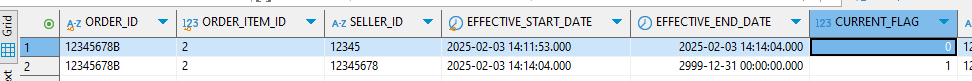
#### Step 4: Update the Record in **STG\_ORDER\_ITEM\_DETAILS**



#### Step 5: Re-run the Informatica Workflow



#### Step 6: Verify the Update in **DIM\_ORDER\_ITEM**



#### Summary of SCD Type 2 Behavior

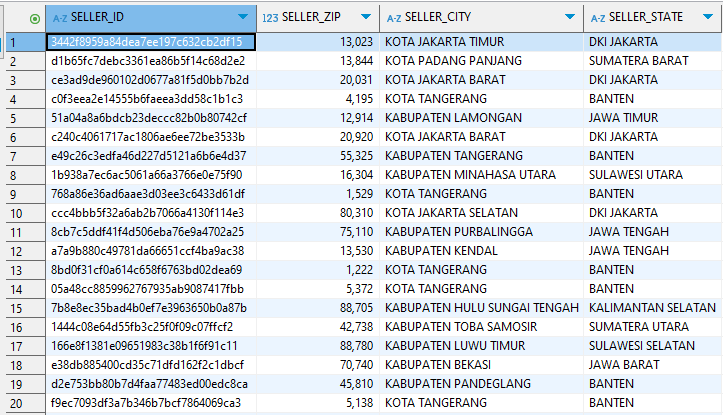
|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
| Insert | Insert a new record in  STG\_ORDER\_ITEM\_DETAILS | The new record appears in DIM\_ORDER\_ITEM with CURRENT\_FLAG =1 |
| Run Workflow | Execute Informatica mapping | Data moves from staging to warehouse |
| Verify Insert | Check DIM\_ORDER\_ITEM | Record exists where  SELLER\_ID=12345 |
| Update | Modify  SELLER\_ID | A new version is created in DIM\_ORDER\_ITEM |
| Run Workflow Again | Re-run Informatica mapping | A new row is inserted, and the old row is marked as historical |
| Verify Update | Check  DIM\_ORDER\_ITEM | NEW Record exists where  SELLER\_ID =12345678 |

## SELLER TABLE

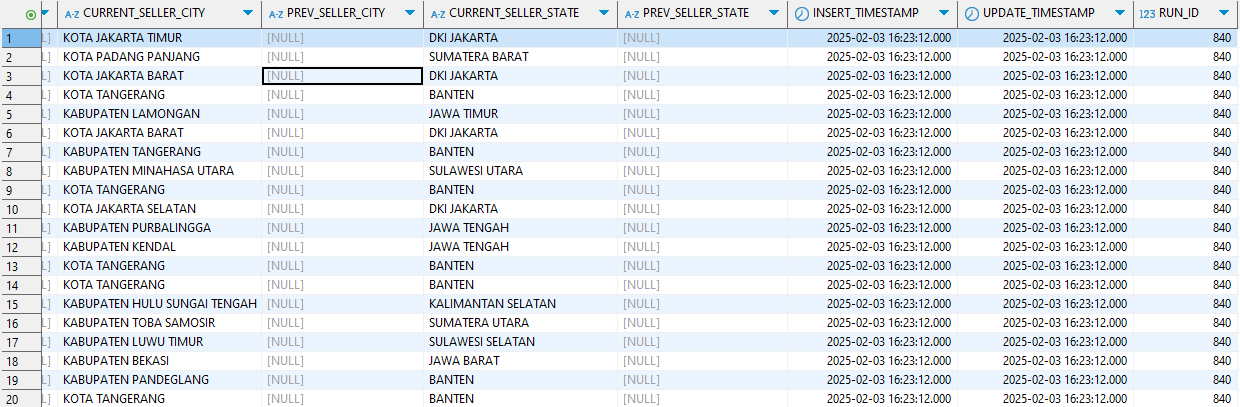
### 1. Mapping Overview(DIM\_SELLER)

* **Mapping Name**: [m\_DIM\_SELLER\_SCD\_T3]
* **Source Table**: [STG\_SELLER\_DETAILS]
* **Target Table**: [DIM\_SELLER]
* **Description**: [ ]

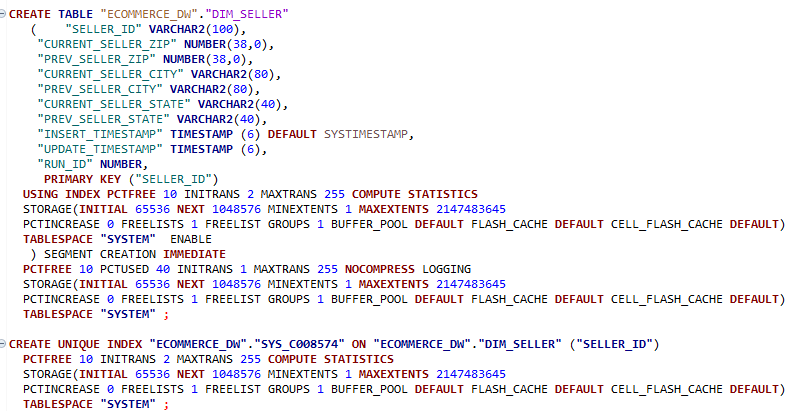
### 2. Source (Staging Table) Details

* **Table Name**: [STG\_SELLER\_DETAILS]
* **Column Structure**:

### 3. Target (Data Warehouse Table) Details

* **Table Name**: [DIM\_SELLER]
* **Table Type**: [Dimension]
* **Column Structure**:

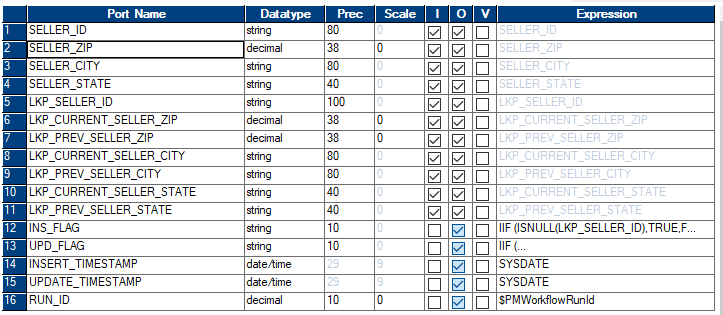
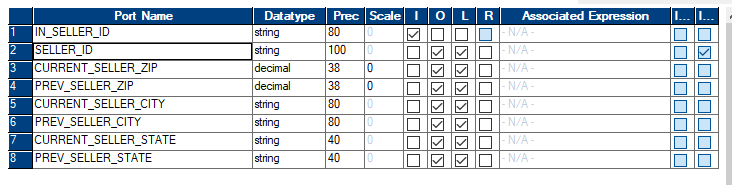
**DDL**

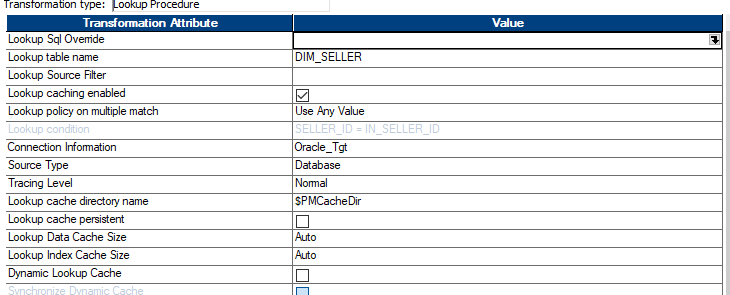
****

* **Primary Key**: [SELLER\_ID)
* **Foreign Keys** : []
* **Indexes**: [ON PRIMARY KEY]

### 4. ETL Mapping Logic

**Transformations Applied**

* **Expression Transformation**
* **Lookup Transformation**

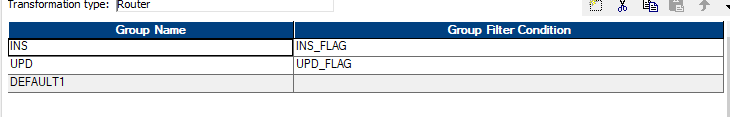
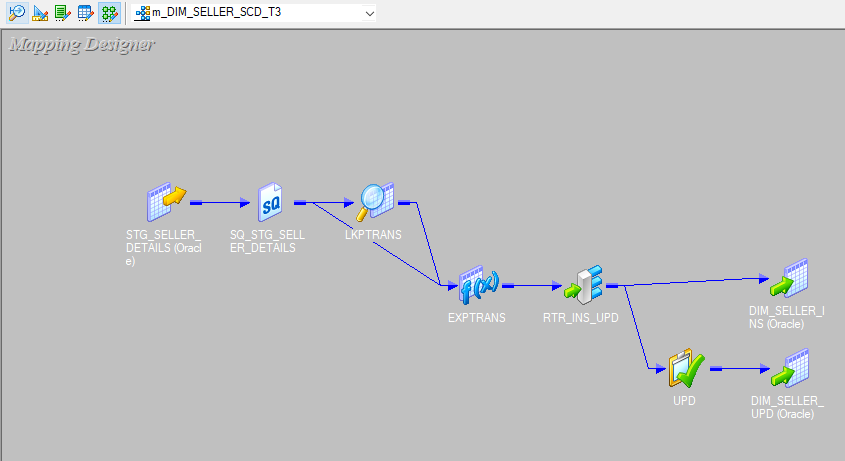


* **Handling Slowly Changing Dimension (SCD) Type 3 in Your Data Warehouse**

**SCD Type 3: Storing Limited Historical Changes**

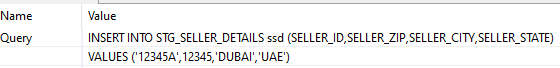
* In **SCD Type 3**, only the **previous value** of a changing attribute is stored alongside the new value.
* This is useful when you only need to track **one level of history** (e.g., previous vs. current address).

Lookup Transformation is **essential** in handling Slowly Changing Dimensions (SCD) because it helps check if a record **already exists** in the target dimension table (DIM\_TABLE). This is necessary to determine whether we should UPDATE OR INSERT:

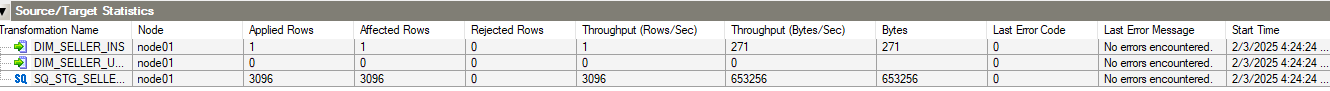
* **ROUTER Transformation**:
* **Router Transformation** is an **active and connected** transformation in Informatica that allows you to route data into multiple output groups based on specified conditions
* WE Use Router Transformation to Separate Updates and Inserts
* **MAPPING**:

### SCD Type 3 Verification Process(DIM\_SELLER)

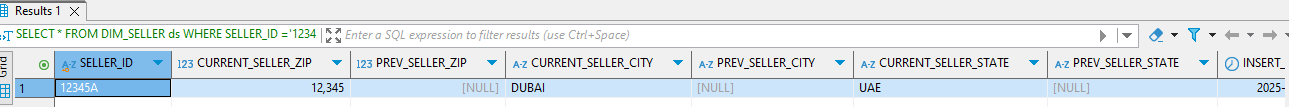
#### Step 1: Insert a New Record into STG\_SELLER\_DETAILS



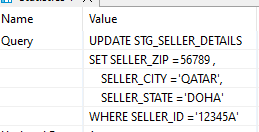
#### Step 2: Run the Informatica Workflow



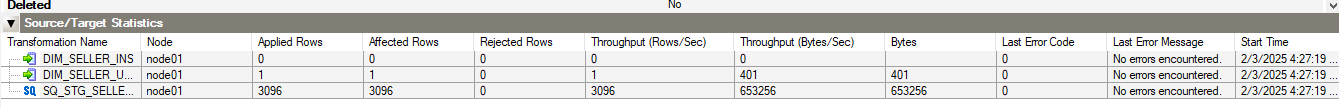
#### Step 3: Verify Data in DIM\_SELLER Table



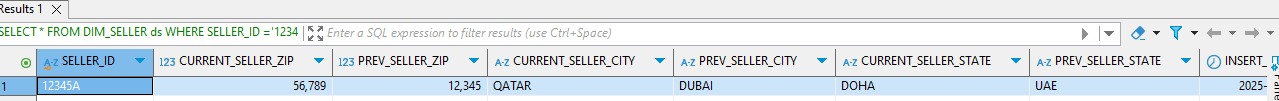
#### Step 4: Update the Record in STG\_SELLER\_DETAILS



#### Step 5: Re-run the Informatica Workflow



#### Step 6: Verify the Update in DIM\_SELLER



### Summary of SCD Type 3 Behavior

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
| Insert | Insert a new record in STG\_SELLER\_DETAILS | The new record appears in DIM\_SELLER |
| Run Workflow | Execute Informatica mapping | Data moves from staging to warehouse |
| Verify Insert | Check DIM\_SELLER | Record exists with CURRENT\_SELLER\_ZIP = 56789 and  PREV\_SELLER\_ZIP =12345 NULL  CURRENT\_SELLER\_CITY= QATAR  PREV\_SELLER\_CITY= NULL  CURRENT\_SELLER\_STATE= DOHA  PREV\_SELLER\_STATE= NULL |
| Update | Modify  CURRENT\_SELLER\_ZIP CURRENT\_SELLER\_CITY  CURRENT\_SELLER\_STATE | The ` PREV\_  stores the old value, and ` CURRENT\_ is updated |
| Run Workflow Again | Re-run Informatica mapping | The new value is updated, keeping history in `PREVIOUS\_ |
| Verify Update | Check DIM\_SELLER | CURRENT\_SELLER\_ZIP = 56789  PREV\_SELLER\_ZIP =12345 CURRENT\_SELLER\_CITY= QATAR  PREV\_SELLER\_CITY= DUBAI CURRENT\_SELLER\_STATE= DOHA  PREV\_SELLER\_STATE= UAE |

## Date Dimension and Time Dimension in Data Warehousing

### 1. Introduction

In a data warehouse, **Date Dimension** and **Time Dimension** play a crucial role in supporting **time-based analysis**. They allow businesses to track trends, measure performance, and analyze events based on different time attributes like **day, week, month, quarter, year, and time of day**.

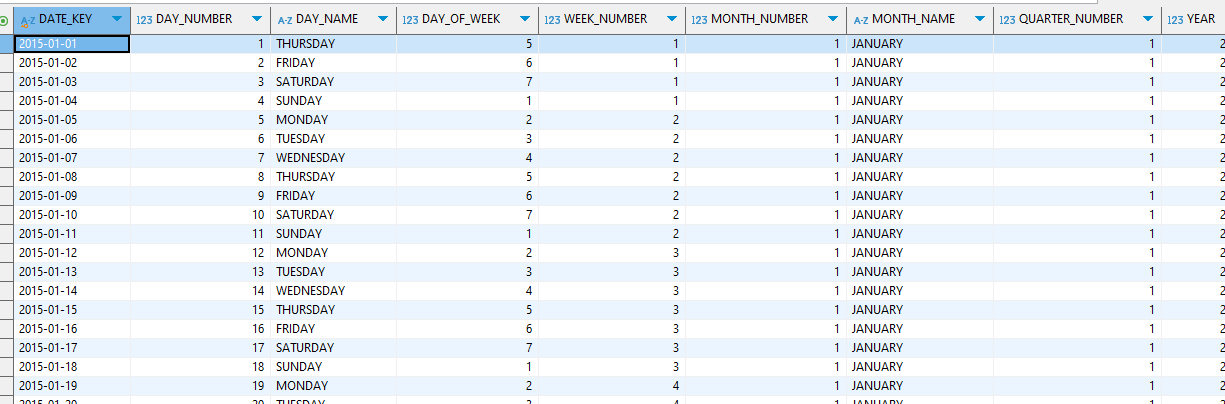
In this project, both dimensions were designed to enable **efficient time-based reporting and querying** in the **data warehouse**.

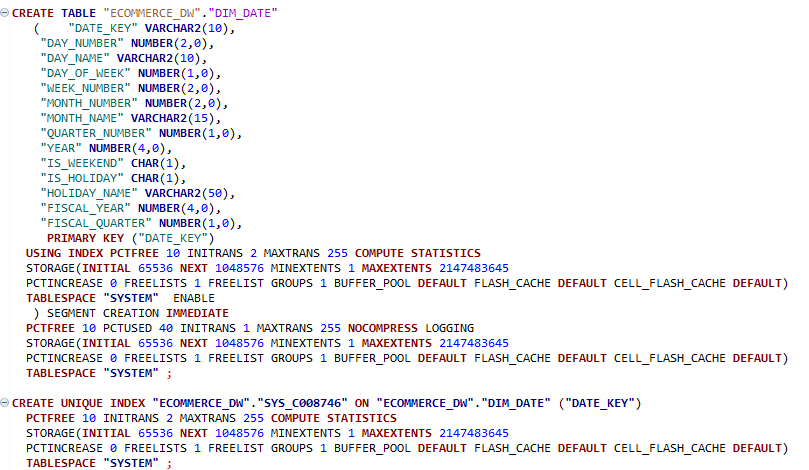
### 2. Date Dimension

**Purpose of the Date Dimension**

The **Date Dimension (DIM\_DATE)** provides a structured calendar that enables users to analyze data over various time periods. It allows filtering, grouping, and aggregating data by **day, week, month, quarter, year, and fiscal periods**.

**Example of Date Dimension Table**

****

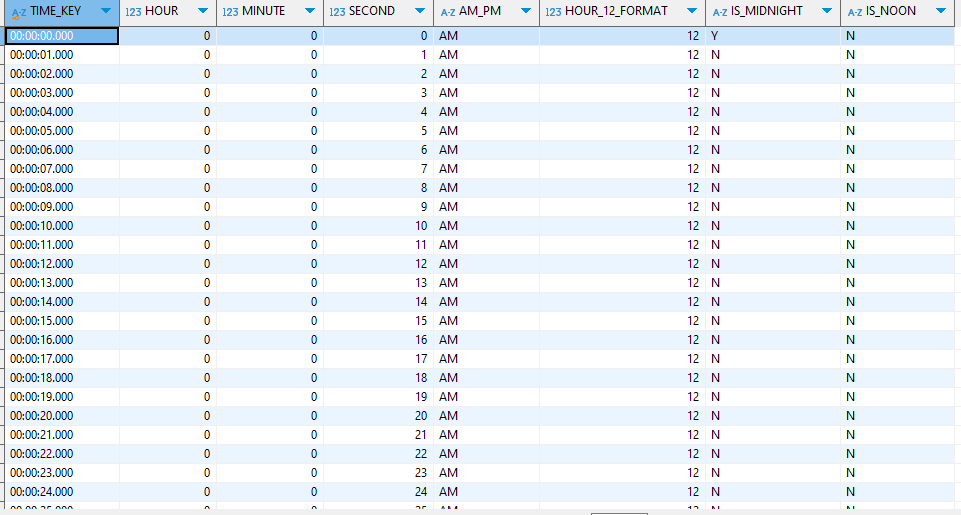
DDL

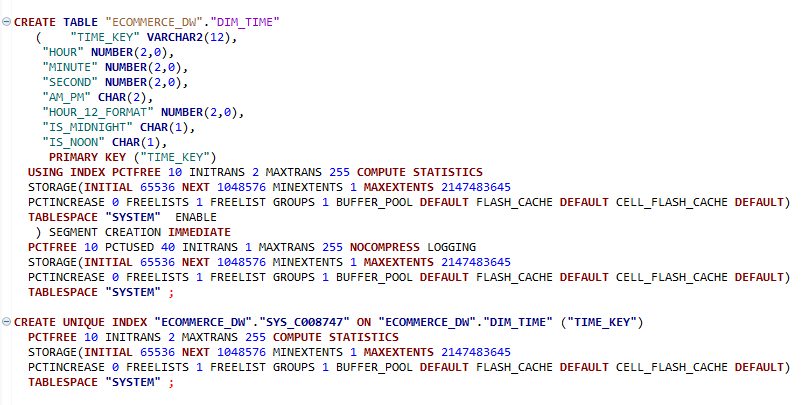
### 3. Time Dimension

**Purpose of the Time Dimension**

The **Time Dimension (DIM\_TIME)** helps analyze events at an **hour, minute, and second** level. It is particularly useful in **transaction-based systems** where time-sensitive tracking is needed.

**Example of Time Dimension Table**



DDL

### 4. How These Dimensions Are Used in Fact Tables

In **fact tables**, timestamps (such as ORDER\_DATE\_TIME) are split into two **foreign key references**:

1. **DATE\_KEY** → References DIM\_DATE.DATE\_KEY, storing the date part.
2. **TIME\_KEY** → References DIM\_TIME.TIME\_KEY, storing the time part.This improves **query performance** and enables **efficient time-based filtering**.