***The Product Company***

**~ Final Data Mart Development Report ~**

Team 5

Team Members

Muriel Banze

Sahil Shah

Siddharth Chauhan

Date

27th November 2020

ISTE-DW Data Warehousing

~ Table of Contents ~

[I. Data Mart Design Definition 2](#_heading=h.30j0zll)

[1.](#_heading=h.1fob9te) Universe of Discourse 2

[2.](#_heading=h.3znysh7) Information Package 2

[II. Dimensional Model 6](#_heading=h.2et92p0)

[III. Data Staging: ETL – Data Extract File Definitions 7](#_heading=h.tyjcwt)

[IV. Data Staging: ETL – Source-to-Target Mappings 14](#_heading=h.3dy6vkm)

[V. SQL Code – Tables & Constraints 15](#_heading=h.1t3h5sf)

[VI. Data Staging Activities - ETL 19](#_heading=h.4d34og8)

[1.](#_heading=h.2s8eyo1) Data Cleansing 19

[2.](#_heading=h.17dp8vu) Data Transformation 22

[3.](#_heading=h.3rdcrjn) Table Population 23

[VII. End User Applications 25](#_heading=h.26in1rg)

[1.](#_heading=h.lnxbz9) Queries 25

[2.](#_heading=h.35nkun2) A View 28

[VIII. Handling Slowly Changing Dimensions (SCD) 39](#_heading=h.1ksv4uv)

# I. Data Mart Design Definition

## 1. Universe of Discourse

| The universe of debate for the data mart is to handle The Product Company's distribution to its consumers across its three branches, namely PEC, TPCE, TPCW.  In order to maximize revenue with lower costs and maintain a stable relationship with vendors, the data mart also handles historical revenues, reports and net profit for all divisions. |
| --- |

## 2. Information Package

Process Name: TPC and its Division's Financial Results.

Grain: Daily Sales for each customer, Product

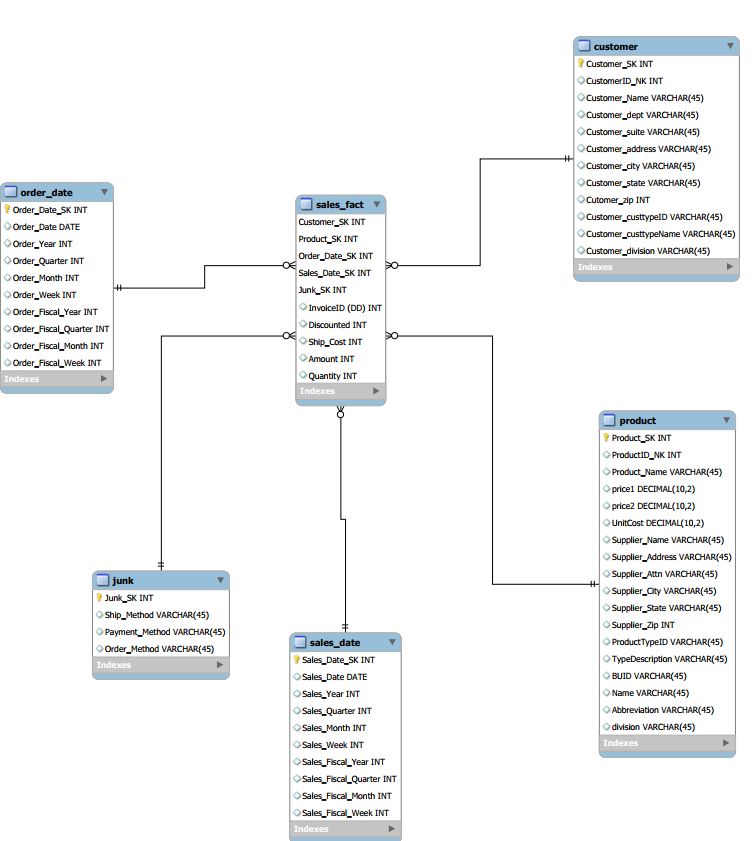
| **Customer** | **Product** | **Order\_Date** | **Sales\_Date** | **Junk** |
| --- | --- | --- | --- | --- |
| Customer\_SK | Product\_SK | Order\_Date\_SK | Sales\_Date\_SK | Junk\_SK |
| CustomerID\_NK | ProductID\_NK | Order\_Date | Sales\_Date | Ship\_Method |
| Customer\_Name | Product\_Name | Order\_Year | Sales\_Year | Payment\_Method |
| Customer\_dept | Price1 | Order\_Quarter | Sales\_Quarter | Order\_Method |
| Customer\_suite | Price2 | Order\_Month | Sales\_Month |  |
| Customer\_address | UnitCost | Order\_Week | Sales\_Week |  |
| Customer\_city | Supplier\_Name | Order\_Fiscal\_Year | Sales\_Fiscal\_Year |  |
| Customer\_state | Supplier\_Address | Order\_Fiscal\_Quarter | Sales\_Fiscal\_Quarter |  |
| Customer\_zip | Supplier\_Attn | Order\_Fiscal\_Month | Sales\_Fiscal\_Month |  |
| Customer\_custtypeID | Supplier\_City | Order\_Fiscal\_Week | Sales\_Fiscal\_Week |  |
| Customer\_custtypeName | Supplier\_State |  |  |  |
| Customer\_division | Supplier\_Zip |  |  |  |
|  | ProductTypeID |  |  |  |
|  | TypeDescription |  |  |  |
|  | BUID |  |  |  |
|  | Name |  |  |  |
|  | Abbreviation |  |  |  |
|  | Division |  |  |  |

Facts: Discounted, Ship\_Cost, Amount, Quantity, InvoiceID (DD)

3. Entity Definitions

| **Entity** | **Entity Definition** (*genus differentia*) |
| --- | --- |
| Customer | The entity holds the all the information of all existing customers in the Organization over all divisions – PEC, TPCW, TPCE.   1. Customer\_SK – All the unique surrogate keys assigned to each customers present in every department. 2. CustomerID\_NK – Natural keys of customers 3. Customer\_Name – Name of the customer 4. Customer\_dept- Department Number of the customer 5. Customer\_suite – Suite Number of the customer 6. Customer\_Address – Address of the Customer 7. Customer\_City- City of the customer in the Address 8. Customer\_State – State where the customers lives 9. Customer\_ZIP – Zipcode of the customer’s Address 10. Customer\_custtypeID – Code of the CustomerType 11. Customer\_custtypeName- Name of the CustomerType 12. Customer\_Division- Defines from which division the customer is from PEC, TPCE or TPCW. |
| Product | The entity holds the all the information of all existing Products in the Organization over all divisions – PEC, TPCW, TPCE   1. Product\_SK – Surrogate Key of Product Dimension 2. ProductID\_NK – Natural Key of Product Dimension 3. Product\_Name- Name of the Product 4. Price1 – Price when discount = 0 5. Price2 – Price when discount = 1 6. UnitCost – Cost of the Unit 7. Supplier\_Name- Name of the Supplier 8. Supplier\_Address – Address of the Supplier 9. Supplier\_Attn – The head representative of the supplier. 10. Supplier\_City – City when the Supplier is located 11. Supplier\_State – State where the Supplier is located 12. Supplier\_ZIP – Zipcode of the Address 13. ProductTypeID – ID of the product type 14. TypeDescription – Description of the Product Type 15. BUID – Business Unit ID 16. Name – Name of the Business Unit 17. Abbreviation – Abbreviation of the Business Unit 18. Division – The division from which companies manufactures and supplies (PEC, TPCE, or TPCW. |
| Order\_Date | It is a role-playing dimension in our dimensional model. It represents order date across all divisions namely (PEC, TPCE, TPCW)   1. Order\_Date\_SK – Surrogate Key of the Order Date 2. Order\_Date – Actual date of the Order(MM/DD/YYYY) 3. Order\_Year- Year of the Order 4. Order\_Quarter- Quarter of the date 5. Order\_Month – Month of the Order 6. Order\_Week – Week of the Order 7. Order\_Fiscal\_Year – Fiscal Year of order 8. Order\_Fiscal\_Quarter – Fiscal Quarter of Order 9. Order\_Fiscal\_Month – Fiscal Month of Order 10. Order\_Fiscal\_Week – Fiscal Week of Order |
| Sales\_Date | It is a role-playing dimension in our dimensional model. It represents sales date across all divisions namely (PEC, TPCE, TPCW)   1. Sales\_Date\_SK – Surrogate key of Sales\_Date 2. Sales\_Date – Actual Sales Date(MM/DD/YYYY) 3. Sales\_Year – Year of the Sales Date 4. Sales\_Quarter – Quarter of Sales Date 5. Sales\_Month – Month of Sales Date 6. Sales\_Week – Week of Sales Date 7. Sales\_Fiscal\_Year – Fiscal Year of Sales Date 8. Sales\_Fiscal\_Quarter – Fiscal Quarter of Sales Date 9. Sales\_Fiscal\_Month – Fiscal Month of Sales Date 10. Sales\_Fiscal\_Week – Fiscal Week of Sales Date |
| Junk | This is a Junk Dimension since attributes are often flag-like in nature as they do not belong to any dimension. This specifies the form of system of delivery, payment and ordering in all dimensions.   1. Junk\_SK – Surrogate Key of Junk 2. Ship\_Method – Method of Shipping 3. Order\_Method – Ordering Method 4. Payment\_Method – Method of Payment |
| Sales\_Fact | The fact table connecting all the dimensional tables.   1. Customer\_SK – The surrogate acting of the Customer, which acts as the Constraint as Foreign and composite primary key in the fact table. 2. Product\_SK – The surrogate key of the product which acts as the foreign and composite primary key in the fact table 3. Order\_Date\_SK - The surrogate key of the Order Date which acts as the foreign and composite primary key in the fact table 4. Sales\_Date\_SK - The surrogate key of the Sales Date which acts as the foreign and composite primary key in the fact table 5. Junk\_SK - The surrogate key of the Junk which acts as the foreign and composite primary key in the fact table 6. Amount – Total Cost of the product 7. Quantity – Quantity of the product that were ordered 8. Discounted – Defines if the product was sold on discounted price or no (0/1) 9. Ship\_Cost – The cost of the shipping 10. InvoiceID(DD) – Acts and the degenerate dimension in the fact table |

# II. Dimensional Model



# III. Data Staging: ETL – Data Extract File Definitions

**Data Source – PEC**

| **Index** | **File\_Name** | **Format** | **Datatype** |
| --- | --- | --- | --- |
| 1 | PECbusiness\_unit.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | BUID - String  NAME - String  ABBREV - String |
| 2 | PECcustomer.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | BUID - String  NAME - String  ABBREV - String |
| 3 | PECcustomer\_type.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | CUSTTYPEID- String  TYPENAME-String |
| 4 | PECinvoice.csv | Fields separated by comma (,) | Invoice-Integer,  Cust-ID-Integer,  salesDate-Date,  prodid-integer,  amt-Integer,  qty-Integer,  shipMethod-String,  shipCost-Decimal,  paymentMethod-String,  orderMethod-String,  orderDate-Date,  discounted-Integer |
| 5 | PECmanufacturingCosts,csv | Fields separated by pipe (|) | Year- Integer,  Month- Integer,  ProdID- Integer,  manufacturingCost- Integer |
| 6 | PECproduct\_type.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;). Rows enclosed by double quotes (“”) | PRODTYPEID- String,  TYPEDESCRIPTION- String,  BUID- String |
| 7 | PECproduct.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;). Rows enclosed by double quotes (“”) | prodid- Integer,  prodDescription- String,  price1- Decimal,  price2- Decimal,  unitCost- Decimal,  supplierName- String,  productTypeID- Integer |

**Data Source – TPCW**

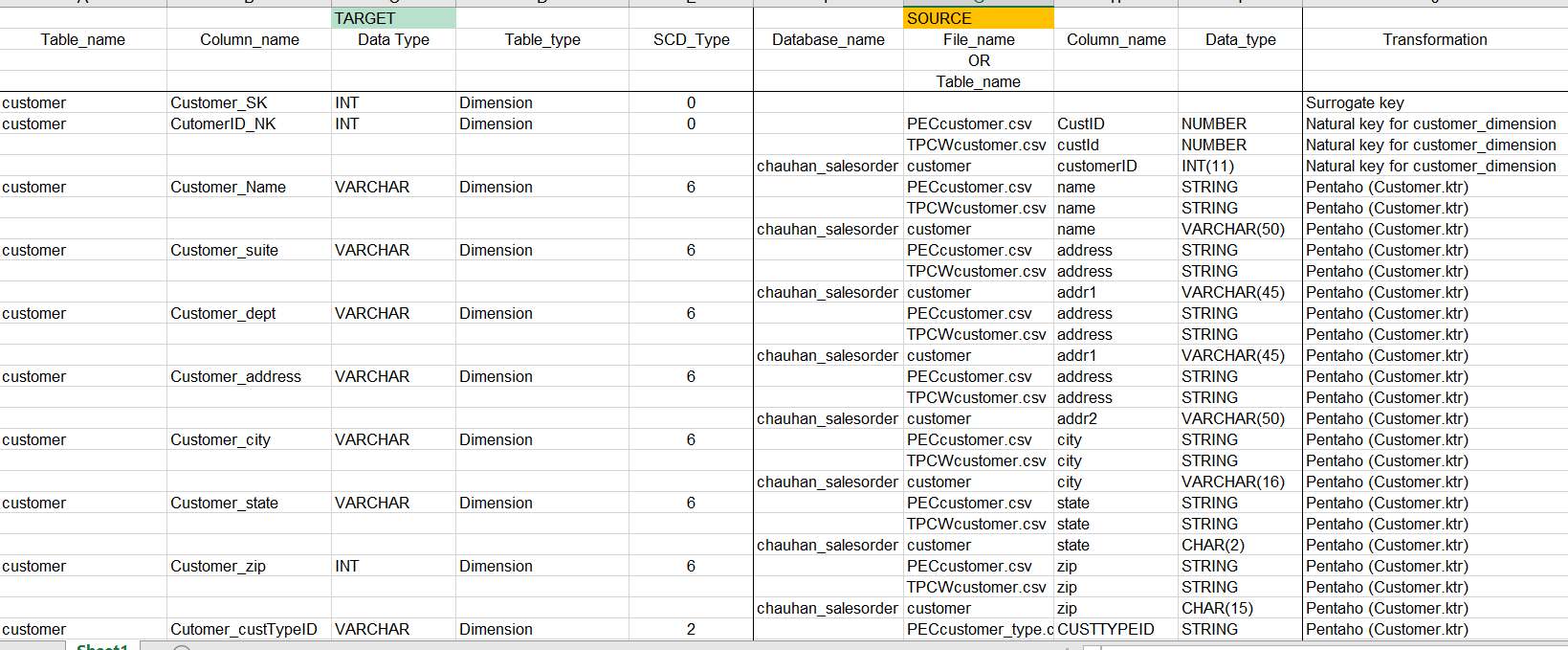
| **Index** | **File\_Name** | **Format** | **Data Type** |
| --- | --- | --- | --- |
| 1 | TPCWbusiness\_unit.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | BUID – String,  NAME- String,  ABBREV-String |
| 2 | TPCWcustomer\_type.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | CUSTTYPEID - String,  TYPENAME-String |
| 3 | TPCWcustomer.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | custID-Integer,  name-String,  address-String,  city-String,  state-String,  zip-Integer,  custType - String |
| 4 | TPCWinvoice.csv | Fields separated by comma (,) | Invoice-Integer,  custID-Integer,  prodID-Integer,  salesDate-String,  amt-Integer,  qty-Integer,  discounted-Integer |
| 5 | TPCWproduct\_type.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;). Rows enclosed by double quotes (“”) | PRODTYPEID- String,  TYPEDESCRIPTION- String,  BUID- String |
| 6 | TPCWproduct.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;). Rows enclosed by double quotes (“”) | ProductID- Integer,  ProductName- String,  Price1- Number,  Price2- Number,  Unit Cost- Number,  Supplier Name- String,  Supplier Address- String,  Supplier city- String,  Supplier State- String,  Supplier zipcode- String,  Product Type ID- Integer |

**Data Source – TPCE**

| **Index** | **File\_Name** | **Format** | **Data Type** |
| --- | --- | --- | --- |
| 1 | business\_unit.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | BUID – String,  NAME- String,  ABBREV-String |
| 2 | invoice.csv | Fields separated by comma (,) | InvoiceID-Integer,  custID-Integer,  salesDate-Date |
| 3 | invoice\_details.csv | Fields separated by comma (,) | InvoiceID – Integer,  prodID- Integer,  amt- Decimal,  qty- Integer,  discounted-Integer |
| 4 | customer\_type.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | CUSTTYPEID - String,  TYPENAME-String |
| 5 | customer.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | CUSTID-Integer,  NAME-String,  ADDR1-String,  ADDR2- String,  CITY-String,  STATE-String,  ZIP-Integer,  CUSTTYPEID-String |
| 6 | supplier.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;). | SUPPLIERID-Integer,  NAME- String,  ADDR1- String,  ADDR2- String,  CITY- String,  STATE- String,  ZIP- Integer |
| 7 | product.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;). | ProductID- Integer,  ProductName- String,  Price1- Number,  Price2- Number,  Unit Cost- Number,  Supplier Name- String,  Supplier Address- String,  Supplier city- String,  Supplier State- String,  Supplier zipcode-String,  Product Type ID- Integer |
| 8 | prod\_type.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;). | PRODTYPEID- String,  TYPEDESCRIPTION- String,  BUID- String |

# IV. Data Staging: ETL – Source-to-Target Mappings

**Reference File : Mapping.xlsx**



# V. SQL Code – Tables & Constraints

**Creating `salesorder\_5\_2201`**

**Query:**

CREATE DATABASE IF NOT EXISTS `salesorder\_5\_2201` /\*!40100 DEFAULT CHARACTER SET utf8 \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

USE `salesorder\_5\_2201`;

**Customer**

**Query:**

DROP TABLE IF EXISTS `customer`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `customer` (

`Customer\_SK` int NOT NULL AUTO\_INCREMENT,

`CustomerID\_NK` int DEFAULT NULL,

`Customer\_Name` varchar(45) DEFAULT NULL,

`Customer\_dept` varchar(45) DEFAULT NULL,

`Customer\_suite` varchar(45) DEFAULT NULL,

`Customer\_address` varchar(45) DEFAULT NULL,

`Customer\_city` varchar(45) DEFAULT NULL,

`Customer\_state` varchar(45) DEFAULT NULL,

`Cutomer\_zip` int DEFAULT NULL,

`Customer\_custtypeID` varchar(45) DEFAULT NULL,

`Customer\_custtypeName` varchar(45) DEFAULT NULL,

`Customer\_division` varchar(45) DEFAULT NULL,

PRIMARY KEY (`Customer\_SK`)

)

**Product**

**Query:**

DROP TABLE IF EXISTS `product`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `product` (

`Product\_SK` int NOT NULL AUTO\_INCREMENT,

`ProductID\_NK` int DEFAULT NULL,

`Product\_Name` varchar(45) DEFAULT NULL,

`price1` decimal(10,2) DEFAULT NULL,

`price2` decimal(10,2) DEFAULT NULL,

`UnitCost` decimal(10,2) DEFAULT NULL,

`Supplier\_Name` varchar(45) DEFAULT NULL,

`Supplier\_Address` varchar(45) DEFAULT NULL,

`Supplier\_Attn` varchar(45) DEFAULT NULL,

`Supplier\_City` varchar(45) DEFAULT NULL,

`Supplier\_State` varchar(45) DEFAULT NULL,

`Supplier\_Zip` int DEFAULT NULL,

`ProductTypeID` varchar(45) DEFAULT NULL,

`TypeDescription` varchar(45) DEFAULT NULL,

`BUID` varchar(45) DEFAULT NULL,

`Name` varchar(45) DEFAULT NULL,

`Abbreviation` varchar(45) DEFAULT NULL,

`division` varchar(45) DEFAULT NULL,

PRIMARY KEY (`Product\_SK`)

)

**Order Date**

**Query:**

DROP TABLE IF EXISTS `order\_date`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `order\_date` (

`Order\_Date\_SK` int NOT NULL AUTO\_INCREMENT,

`Order\_Date` date DEFAULT NULL,

`Order\_Year` int DEFAULT NULL,

`Order\_Quarter` int DEFAULT NULL,

`Order\_Month` int DEFAULT NULL,

`Order\_Week` int DEFAULT NULL,

`Order\_Fiscal\_Year` int DEFAULT NULL,

`Order\_Fiscal\_Quarter` int DEFAULT NULL,

`Order\_Fiscal\_Month` int DEFAULT NULL,

`Order\_Fiscal\_Week` int DEFAULT NULL,

PRIMARY KEY (`Order\_Date\_SK`)

)

**Sales Date**

**Query:**

DROP TABLE IF EXISTS `sales\_date`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `sales\_date` (

`Sales\_Date\_SK` int NOT NULL AUTO\_INCREMENT,

`Sales\_Date` date DEFAULT NULL,

`Sales\_Year` int DEFAULT NULL,

`Sales\_Quarter` int DEFAULT NULL,

`Sales\_Month` int DEFAULT NULL,

`Sales\_Week` int DEFAULT NULL,

`Sales\_Fiscal\_Year` int DEFAULT NULL,

`Sales\_Fiscal\_Quarter` int DEFAULT NULL,

`Sales\_Fiscal\_Month` int DEFAULT NULL,

`Sales\_Fiscal\_Week` int DEFAULT NULL,

PRIMARY KEY (`Sales\_Date\_SK`)

)

**Junk**

**Query:**

DROP TABLE IF EXISTS `junk`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `junk` (

`Junk\_SK` int NOT NULL AUTO\_INCREMENT,

`Ship\_Method` varchar(45) DEFAULT NULL,

`Payment\_Method` varchar(45) DEFAULT NULL,

`Order\_Method` varchar(45) DEFAULT NULL,

PRIMARY KEY (`Junk\_SK`)

)

**Sales Fact**

**Query:**

DROP TABLE IF EXISTS `sales\_fact`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `sales\_fact` (

`Customer\_SK` int NOT NULL,

`Product\_SK` int NOT NULL,

`Order\_Date\_SK` int NOT NULL,

`Sales\_Date\_SK` int NOT NULL,

`Junk\_SK` int NOT NULL,

`InvoiceID (DD)` int DEFAULT NULL,

`Discounted` int DEFAULT NULL,

`Ship\_Cost` decimal(10,2) DEFAULT NULL,

`Amount` decimal(10,2) DEFAULT NULL,

`Quantity` int DEFAULT NULL,

PRIMARY KEY (`Customer\_SK`,`Product\_SK`,`Order\_Date\_SK`,`Sales\_Date\_SK`,`Junk\_SK`),

KEY `product\_fk\_idx` (`Product\_SK`),

KEY `salesDate\_fk\_idx` (`Sales\_Date\_SK`),

KEY `orderDate\_fk\_idx` (`Order\_Date\_SK`),

KEY `junk\_fk\_idx` (`Junk\_SK`),

KEY `customer\_fk\_idx` (`Customer\_SK`),

CONSTRAINT `customer\_fk` FOREIGN KEY (`Customer\_SK`) REFERENCES `ustomer` (`Customer\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `junk\_fk` FOREIGN KEY (`Junk\_SK`) REFERENCES `junk` (`Junk\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `orderDate\_fk` FOREIGN KEY (`Order\_Date\_SK`) REFERENCES `order\_Date` (`Order\_Date\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `product\_fk` FOREIGN KEY (`Product\_SK`) REFERENCES `product` (`Product\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `salesDate\_fk` FOREIGN KEY (`Sales\_Date\_SK`) REFERENCES `sales\_Date` (`Sales\_Date\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION

)

# VI. Data Staging Activities - ETL

## 1. Data Cleansing

| **DM Table** | **Attribute** | **Problem** | **Resolution Strategy** (attach code) |
| --- | --- | --- | --- |
| **Customer** | | | |
| PEC\_Customer | Custtype  Address | * Extra Commas * Removing periods and field names with half names. For eg: St.,Av.,Dr.,Rd   Changing Uppercase to lower case for custtype | * Replace in String function   Use Constant to a define suite/dept as N/A as they are not present. |
| PEC\_Custtype | CusttypeID  TypeName | Extra Inverted Commas | Replace in String with setting as empty string. |
| TPCW\_Customer | All fields | * Extra Inverted Commas * Partial names for custtype field with Edu., Comm., Govt, State., * Partial names in name with Inc., Co, Corp. * Partial names in Address field St, Rd, Ave.   Lowercase in State with Fl and DC | * Replace in string for every field where the problem exists. |
| TPCE\_Customer | Address | Dept and Suite in address field |  |
| TPCE\_Custtype | typeName | (\_) in State/Local Govt | Replace in String function to (/) |
| **Product** | | | |
| PEC\_product | All fields | * Replace with inverted commas   Unit cost are calculated incorrectly. | * Replace in String and Equip to Equipment * We merge the Product and manufacturing cost based on the Sales-date with respect to month and year.   UnitPrice = manufactoringCost/TotalQty. |
| PEC\_Producttype | All fields | Replace with Inverted Commas | Replace is String and Equip to Equipment |
| PEC\_BusinessUnit | All fields | * Replace with Inverted Commas   Abbreviation Missing in Miscellaneous | * Replace in String |
| PEC\_manufacturingCost | Year | The year consist of only YY |  |
| TPCW\_businessUnit | Abbrev | Abbreviation Missing in Miscellaneous |  |
| TPCW\_prodtype | All fields | Extra Inverted Commas | Replace in String with Set to Empty String |
| TPCW\_Product | All fields | * Extra Inverted Commas * Suite and Dept in Same Address field   Lowercase Supplier State | * Replace in String with Set to Empty String * Splitting Fields function |
| TPCE\_BusinessUnit | Abbrev | Abbreviation Missing in Miscellaneous |  |
| **Order\_Date** | | | |
| PEC\_invoice | Order\_date | Two records had black values, because shifting of columns | Used Modified Javascript to replace the values. Other approach was to make changes manually in excel. |
| **Sales\_Date** | | | |
| PEC\_invoice | saleDate | Incorrect Date and format of date | * Segregating the dates in to year, Month and week.   Select values to change the date format to MMddYYYY |
| TPCE\_invoice | saleDate | Date in ddMMYYYY | Used Modified Java Script |
| TPCW\_invoice | salesDate | Inconsistencies in date with different formats ddmmyy and ddMMYYYY | * Used Microsoft Excel to make the changes. |
| **Junk** | | | |
| PEC\_invoice | shipMethod | Incorrect names for air, train, truck. |  |

## 2. Data Transformation

| **DM Table** | **Image Creation Process** (attach code) |
| --- | --- |
| Customer | * Getting Input from all the sources of the company – PEC, TPCW and TPCE after cleaning them. * After the cleaning process, we merged them giving each one of them division to identify from where the customers are coming from i.e. PEC, TPCW, TPCE. * Transformations are sorted from PEC, TPCE and TPCW. * We assigned the Surrogate keys to each unique records. |
| Product | * Cleaning all the product files from PEC, TPCE and TPCW we assign the constants for each division. * Merging them with the feature of Pentaho “Append Streams”, first with PEC and TPCW and Later TPCE. * We didn’t split the supplier and kept inside the product dimension with name, address and state. * Added Surrogate keys to unique each unique records in Product dimension. |
| Order\_date | * PEC product dates helped to calculate the Order\_Year, Order\_Month, Order\_Week, Order\_Quarter. * Removed duplicates using unique rows. * Added new fields in the Order\_date dimension with fiscal\_year, fiscal\_Quarter, fiscal\_Month and fiscal\_week using javascript. * Added surrogate keys for each unique dates |
| Sales\_Date | * Using calculator got the separate field for Sales\_Year, Sales\_Quarter, Sales\_Month and Sales\_Week for all the divisions (PEC, TPCW and TPCE) * Same as order date used javascript to get the fiscal\_year, fiscal\_Quarter, fiscal\_Month and fiscal\_week. * Added surrogate keys for each unique date. |
| Junk | * PEC invoice only has orderMethod, PaymentMethod and shipMethod so extracted from the invoice. * Added orderMethod, PaymentMethod and shipMethod to TPCE and TPCW with “N/A”. * Sorted the rows with each unique rows to remove duplicates. |

## 3. Table Population

| **DM Table** | **Table Population Process** (attach code) |
| --- | --- |
| Customer |  |
| Product |
| Order\_Date |
| Junk |
| Sale\_Date |
| Sales\_Fact |  |
|  | * Merging each tables using Database Lookup with surrogate keys of Customer, Product, Sales\_Date, Order\_Date and Junk. * Matching the records based on Natural Keys and Division in similarity. * Group by the records for redundant records of the same SK for the population of the fact table. |

# VII. End User Applications

## 

## 1. Queries

| **User Question/Reporting Need** |
| --- |
| **Report that is showing Top 5 Customers of Each Division with customer\_Type, Product description, name and total sales amount.**   1. **PEC** 2. **TPCE** 3. **TPCW** |
| **SQL Code** |
| **1.** select c.Customer\_Name, c.Customer\_custtypeName, p.Product\_Name, p.TypeDescription, c.Customer\_division, SUM(Amount) as Sales from sales\_fact AS sf  join customer AS c using(Customer\_SK)  join product as p using(Product\_SK)  where c.Customer\_division = "PEC"  group by c.Customer\_Name, C.Customer\_state  order by sum(Sales) DESC LIMIT 5;    **2. select c.Customer\_Name, c.Customer\_custtypeName, p.Product\_Name, p.TypeDescription, c.Customer\_division,**  **SUM(Amount) as Sales from sales\_fact AS sf**  **join customer AS c using(Customer\_SK)**  **join product as p using(Product\_SK)**  **where c.Customer\_division = "TPCE"**  **group by c.Customer\_Name, C.Customer\_state**  **order by sum(Sales) DESC LIMIT 5;**    **3.** select c.Customer\_Name, c.Customer\_custtypeName, p.Product\_Name, p.TypeDescription, c.Customer\_division,  SUM(Amount) as Sales from sales\_fact AS sf  join customer AS c using(Customer\_SK)  join product as p using(Product\_SK)  where c.Customer\_division = "TPCW"  group by c.Customer\_Name, C.Customer\_state  order by sum(Sales) DESC LIMIT 5; |
| **Supporting Index(es)** |
| **Customer\_Name, Customer\_custtypeName, Customer\_Division(Customer), Product\_Name, TypeDescription(Product), Sales(sales\_fact)** |

| **User Question/Reporting Need** |
| --- |
| **Get the report for most frequent method of ordering a product from PEC division** |
| **SQL Code** |
| SELECT jk.Order\_Method, count(\*) 'Frequency'  FROM sales\_fact sf JOIN junk jk USING (Junk\_SK)  JOIN product p USING (Product\_SK)  WHERE p.division = "PEC"  GROUP BY jk.Order\_Method  ORDER BY Frequency DESC |
| **Supporting Index(es)** |
| **Order\_Method(Junk)** |

| **User Question/Reporting Need** |
| --- |
| **Report where total percentage is calculated for each payment method i.e. COD, Charge and Cash from PEC.** |
| **SQL Code** |
| SELECT a.pm 'Payment Method', a.totaleachinvoices 'Total of Each Invoices', b.total 'Total Invoices',  format(100\*a.totaleachinvoices/b.total,1) "Percentage"  FROM (SELECT j.Payment\_Method 'pm', COUNT(\*) 'totaleachinvoices'  FROM sales\_fact sf  JOIN junk j USING(Junk\_SK)  GROUP BY j.Payment\_Method) a,  (SELECT COUNT(\*) 'total'  FROM sales\_fact) b  LIMIT 3; |
| **Supporting Index(es)** |
| **Payment\_Method(Junk)** |

| **User Question/Reporting Need** |
| --- |
| **Report that defines the total sales with respect to Customer Type and respective Sales Year.** |
| **SQL Code** |
| Select Customer\_custtypeName, Sales\_Year, Sales\_Quarter, Sales\_Month, SUM(Amount) as Sales  FROM sales\_fact as sf  JOIN sales\_date as sd USING(Sales\_Date\_SK)  JOIN customer as cd USING (Customer\_SK)  GROUP BY Customer\_custtypeName, Sales\_Year, Sales\_Quarter  ORDER BY SUM(Amount) DESC; |
| **Supporting Index(es)** |
| **Customer\_typeName(Customer), Sales\_Year, Sales\_Quarter,Sales\_Month(Sales\_Date), Amount(sales\_fact)** |

## 

## 2. A View

**The view reports the sales report for each year, sales, cost and calculate the profit with respect to division to see who did good in all those years.**

1. **PEC**

**Query for the view:**

CREATE VIEW PEC\_gross\_profit\_year AS

SELECT view1.division, view1.Year, view1.Sales, view1.Costs, (view1.Sales-view1.Costs) 'Gross Profit'

FROM

(SELECT p.division, s.Sales\_Year 'Year', SUM(salesfact.Amount) 'Sales', SUM(p.UnitCost \* salesfact.Quantity) 'Costs'

FROM sales\_fact salesfact JOIN product p USING (Product\_SK)

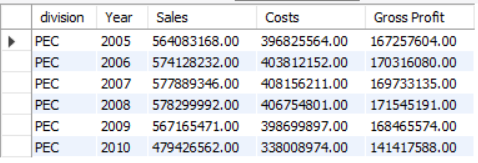
JOIN sales\_date s USING (Sales\_Date\_SK)

WHERE p.division = "PEC"

GROUP BY s.Sales\_Year) view1;

**Query:**

select \* from PEC\_gross\_profit\_year;



1. **TPCE**

**Query for the view:**

CREATE VIEW TPCE\_gross\_profit\_year AS

SELECT view2.division, view2.Year, view2.Sales, view2.Costs, (view2.Sales-view2.Costs) 'Gross Profit'

FROM

(SELECT p.division, s.Sales\_Year 'Year', SUM(salesfact.Amount) 'Sales', SUM(p.UnitCost \* salesfact.Quantity) 'Costs'

FROM sales\_fact salesfact JOIN product p USING (Product\_SK)

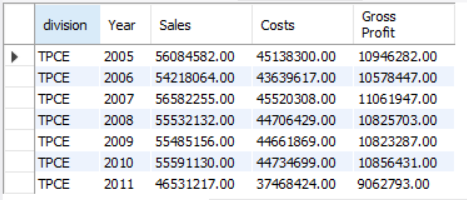
JOIN sales\_date s USING (Sales\_Date\_SK)

WHERE p.division = "TPCE"

GROUP BY s.Sales\_Year) view2;

**Query:**

select \* from TPCE\_gross\_profit\_year;



1. **TPCW**

**Query for the view:**

CREATE VIEW TPCW\_gross\_profit\_year AS

SELECT view3.division, view3.Year, view3.Sales, view3.Costs, (view3.Sales-view3.Costs) 'Gross Profit'

FROM

(SELECT p.division, s.Sales\_Year 'Year', SUM(salesfact.Amount) 'Sales', SUM(p.UnitCost \* salesfact.Quantity) 'Costs'

FROM sales\_fact salesfact JOIN product p USING (Product\_SK)

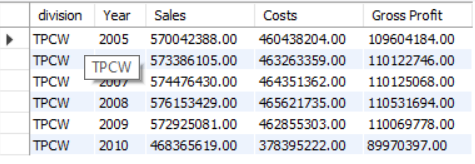
JOIN sales\_date s USING (Sales\_Date\_SK)

WHERE p.division = "TPCW"

GROUP BY s.Sales\_Year) view3;

**Query:**

select \* from TPCW\_gross\_profit\_year;



1. **Overall all the divisions**

CREATE VIEW alldivisions\_gross\_profit\_year AS

SELECT view4.division, view4.Year, view4.Sales, view4.Costs, (view4.Sales-view4.Costs) 'Gross Profit'

FROM

(SELECT p.division, s.Sales\_Year 'Year', SUM(salesfact.Amount) 'Sales', SUM(p.UnitCost \* salesfact.Quantity) 'Costs'

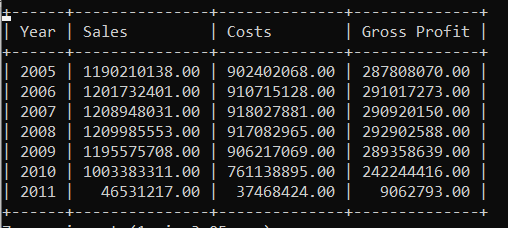
FROM sales\_fact salesfact JOIN product p USING (Product\_SK)

JOIN sales\_date s USING (Sales\_Date\_SK)

GROUP BY s.Sales\_Year) view4;

**Query:**

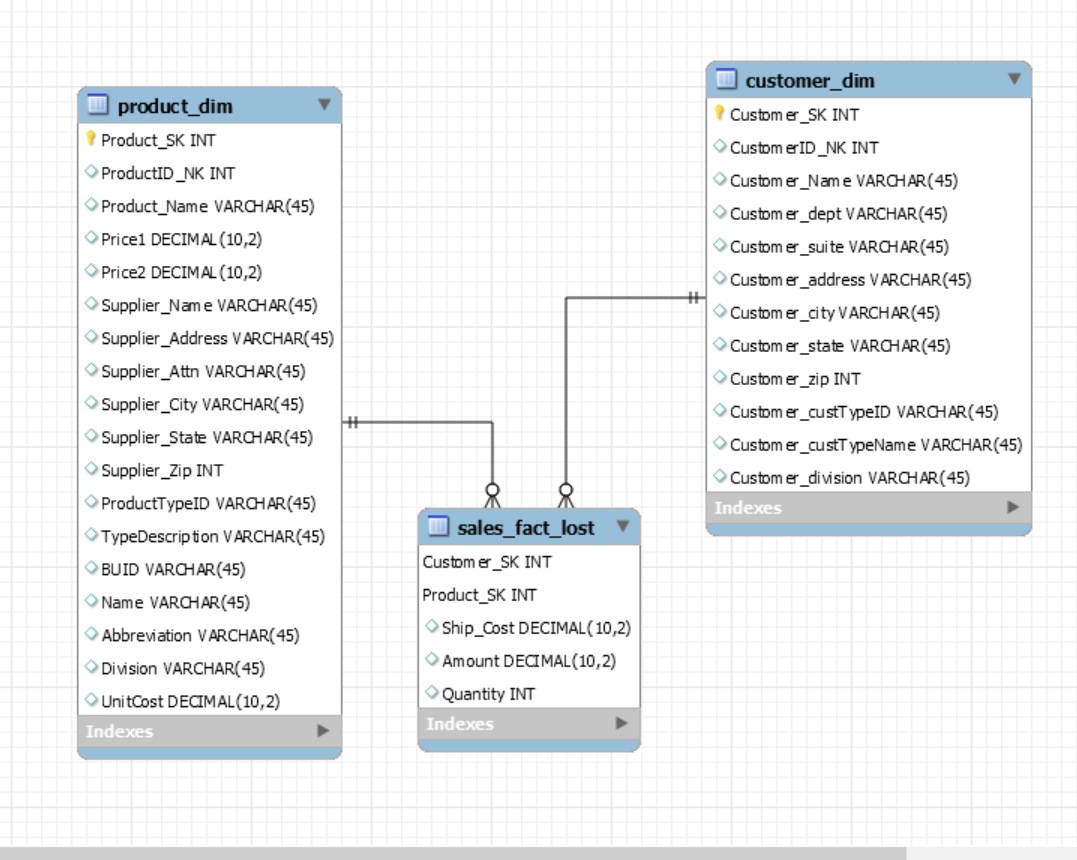
select \* from alldivisons\_gross\_profit\_year;



3. Aggregated Mata Marts

* **Lost** **Aggregated Data Mart**

**ERD:**



**Aggregation method**: Lost Dimension

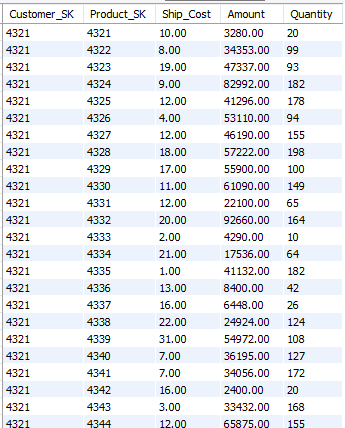
**Use Case**: In this use case we kept Customer and Product dimension. In lost aggregated dimension, one or more dimensions are removed, in this use case we removed sales date, Order date, Junk Dimension. The remaining dimensions can be used in finding the most popular product and the top customers.

**Creation of Tables:** For this kind of aggregation we created the tables using MySQL Workbench and used “Forward Engineering” to further create the data mart

**File:** LostAggregation.mwb

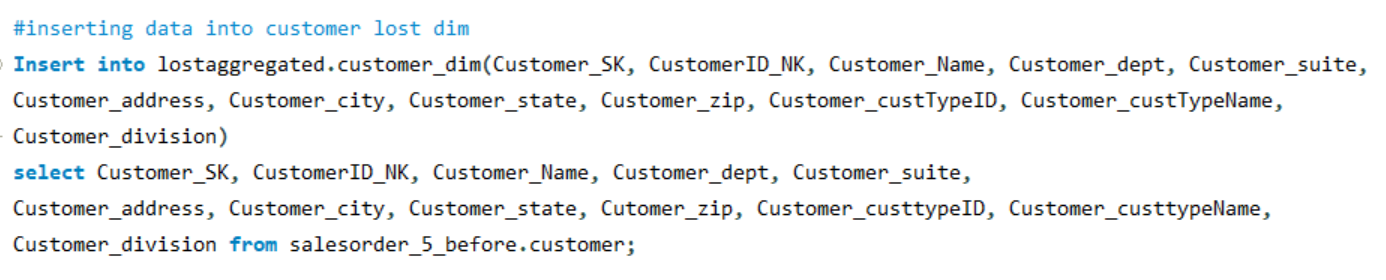
**Output of Fact Table:**



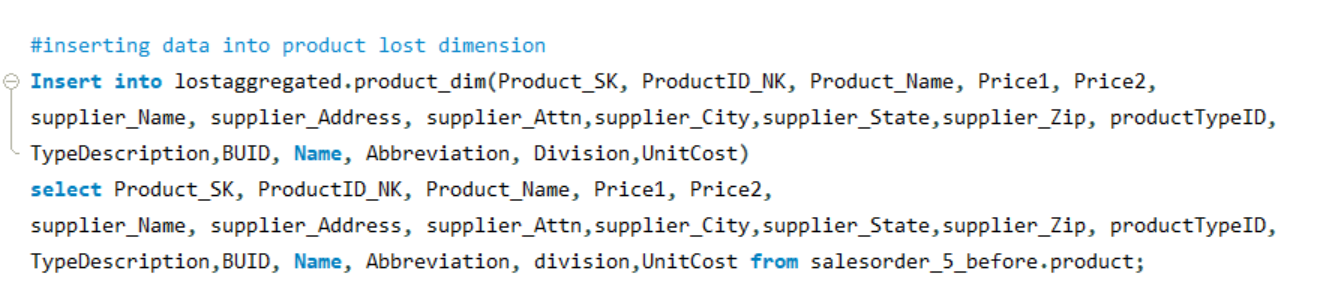


**Creation and Inserting to Aggregates data mart:**

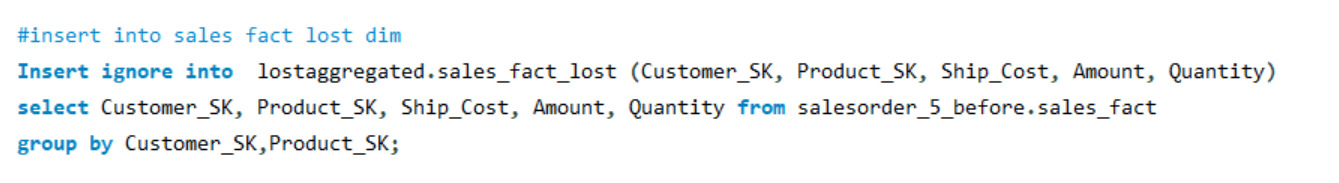
**Customer Dimension:**



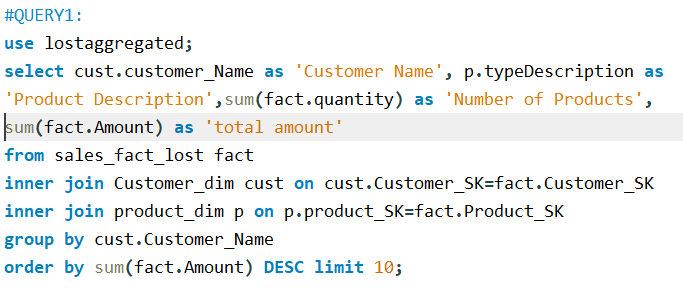
**Product Dimension:**

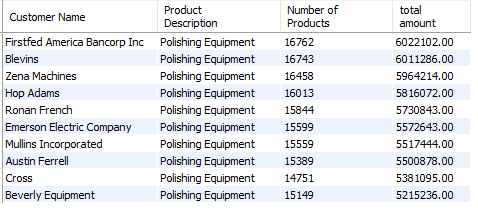


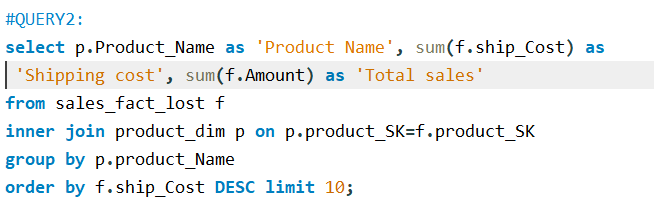
**Lost Aggregation Fact Table:**

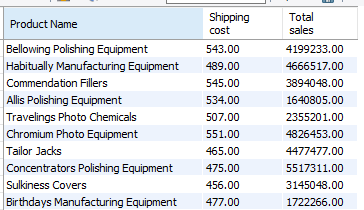


**USER QUERIES:**









* **Shrunken Aggregation data :**

**ERD:**

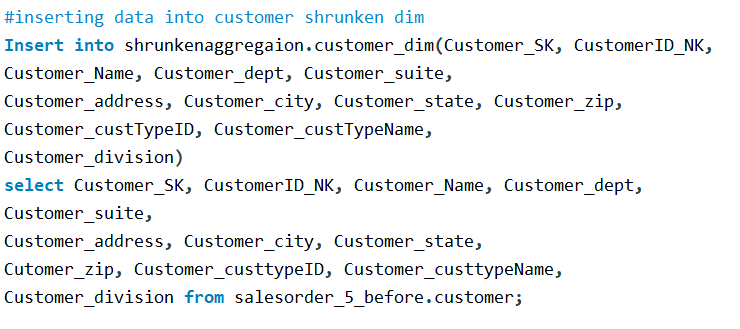
**Aggregation Method:** Shrunken Dimension

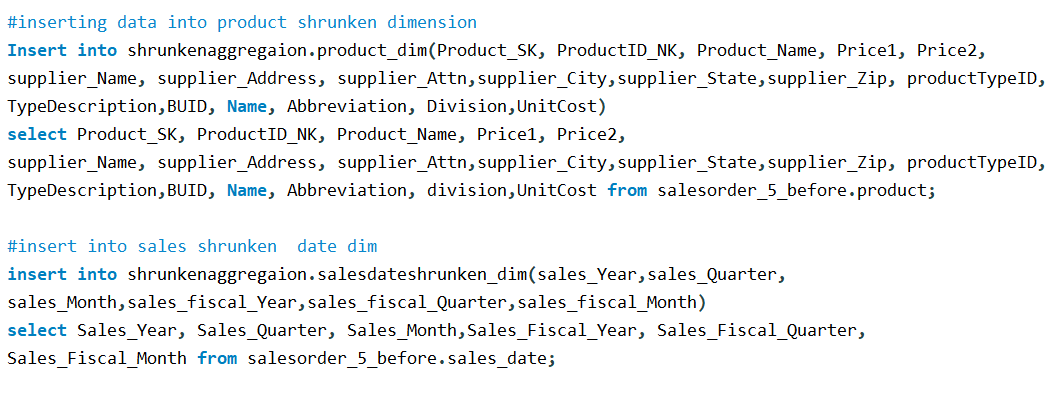
**Use Case:** In this use case we have used the shrunk dimension and made the few dimension lost

**Creation of Tables:** For this kind of aggregation we created the tables using MySQL Workbench and used “Forward Engineering” to further create the data mart

**File:** shrunkenAggregate.mwb

Shruken Dimension & Grain: Monthly



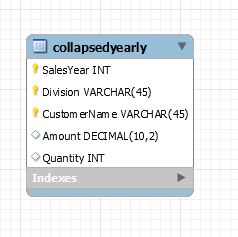


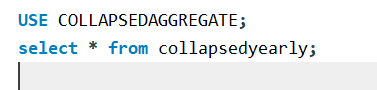
* **Collapsed Aggregation**

**Aggregation Method:** Collapsed Aggregation Dimension

**Use Case:**  In collapsed aggregated Data mart we removed the surrogate key and use the attributes of useful dimensions and combines them into fact table. Here we used sales\_year from sales\_date dimension, Division and CustomerName from Customer dimension.

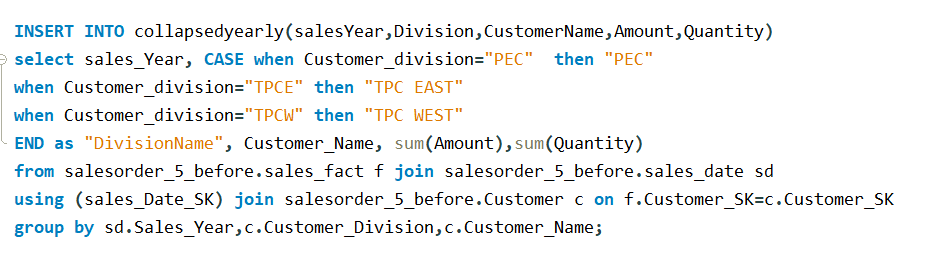
**Sample output:**



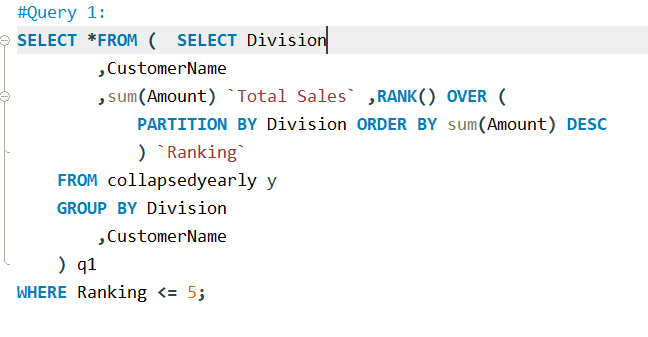


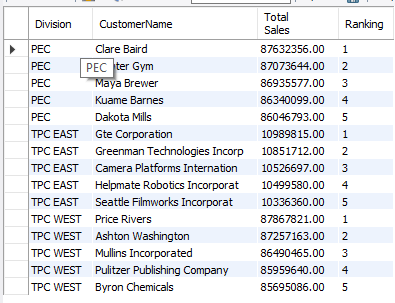


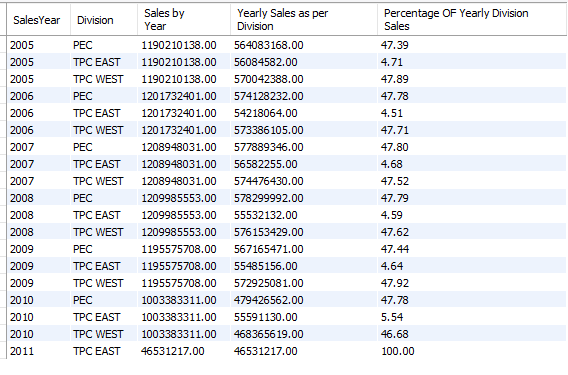
**Populating Collapsed Aggregate Data Mart:**



**USE QUERIES:**





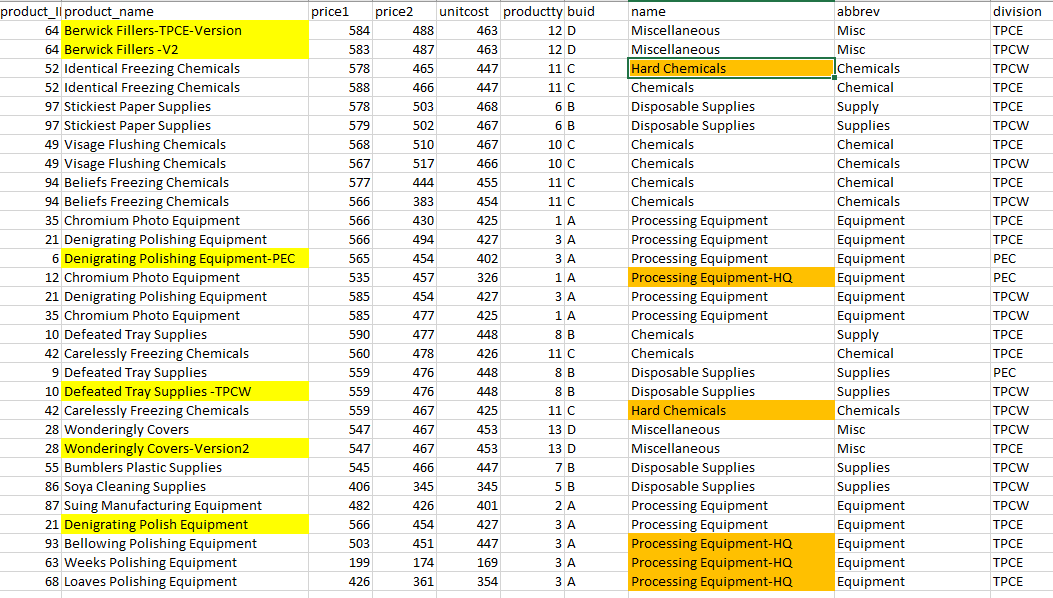


# VIII. Handling Slowly Changing Dimensions (SCD)

* **Created Sample Slowly Changing Dimension records in Product and Customer dimension**

**Source File : SCD\_Data/SCD1\_Data**

**Rows : 30**



* **Highlighted Parts:**

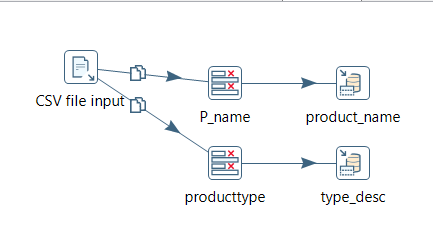
Product\_Name (Yellow) : SCD1 Implementation

Name(Orange): SCD1 Implementation

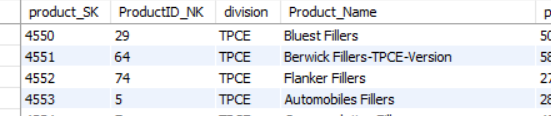
* **SCD Type 1:**

The implantation of the type1 on columns Product\_name and name from Product table as history is not necessary to be kept as it is and it can change overtime.

We implemented SCD1 using the Pentaho, the file is SCD1.ktr, where we used the feature *Output->Insert/Update to implement type1*



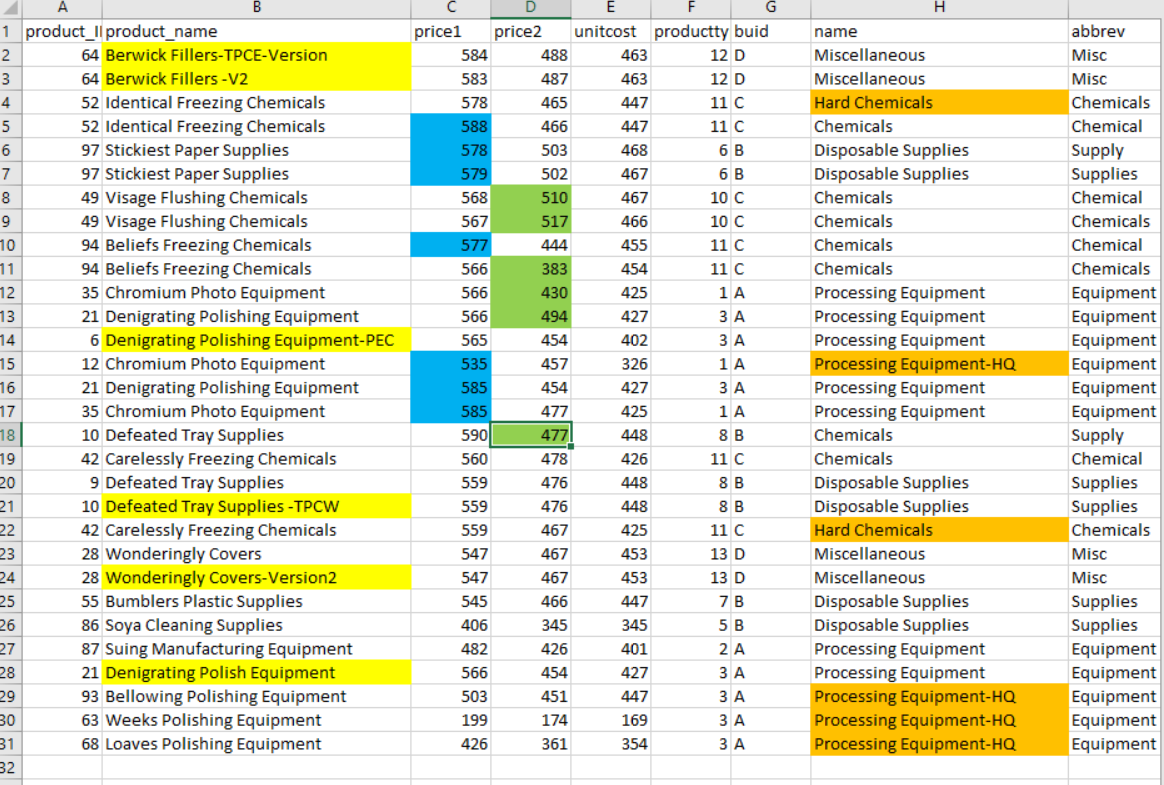
* **Output:**



**SCD Type 2**

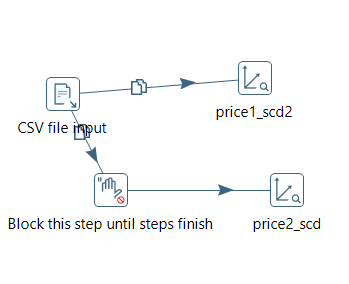
**Source : SCD1\_Data**

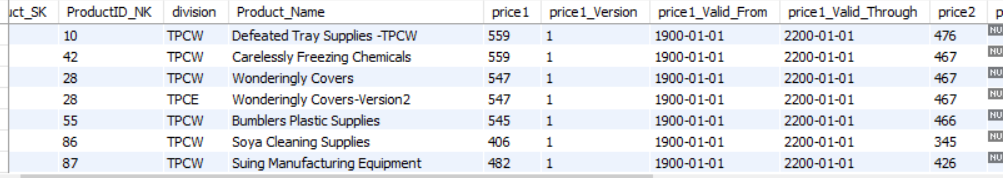
**Rows:30**



* **Highlighted Part:**
  + Price1 (Blue) : SCD type 2
  + Price2 (Green) : SCD type 2

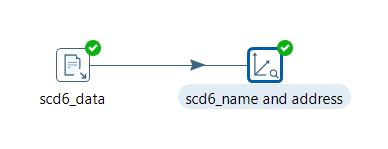
Implementing the price1 and price2 to keep the track of the changes in the prices with their version numbers. Implemented using the *Dimension lookup/Update* for Type2.

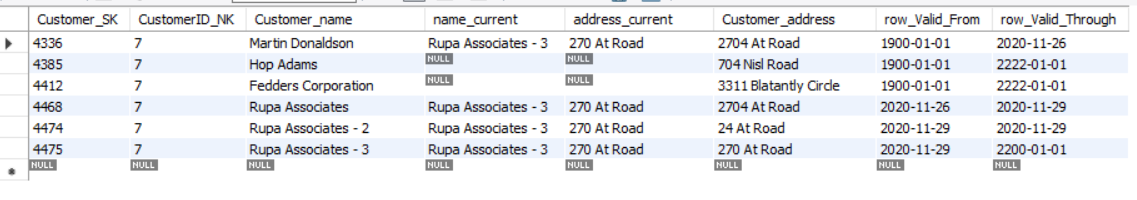




**SCD Type 6**

We implemented SCD Type-6 on the columns name and address from the customer\_dimension. We did so because the history of these attributes is important and should be maintained. Also, Type 6 SCD adds a current field which helps us to determine the current record and the date it is valid till. We implemented SCD-6 using Pentaho transformation (scd6.ktr). In the transformation, we used the step *Data Warehouse → Dimension lookup/ update* to implement SCD Type-6.





IX. Many-to-Many (N-M) Relationship Implementation Option

A Many-to-Many relationship is defined as a relationship between a parent and the child tables in a database. A parent row consists of multiple child rows in the other table. In a relational database design, many-to-many relationships are not permitted due to the following issues:

1. Causes data redundancy
2. Difficulty in inserting, updating, and deleting the data.

In many real-world applications N:M relationships are often used and normalizing the fact table is not an option.

Several approaches exist in dealing Many-to-Many relationships such as:

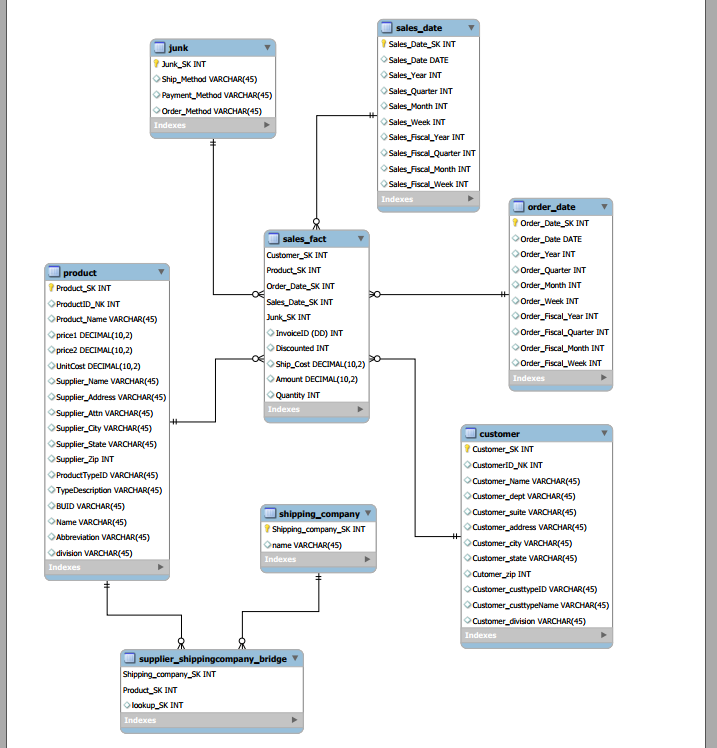
1. Joining or Bridging tables.
2. Lowering the grain of the fact table
3. De-normalizing the Dimension Table by Positional-Flag Attributes
4. De-normalizing the Dimension Table by Non-Positional attributes & a Concatenated Field

For our model we have decided to implement the Bridge method. Here a look-up table is created which consists of the surrogate keys for both the Product and Shipping company.

We intent to find details of the supplier which in our case is present in the Product dimension. Here, many-to-many relationship can occur since there is a possibility of suppliers have multiple shipping contracts. Likewise, the shipping company can have multiple suppliers. The Kimball’s method is a better approach since it minimizes redundancy. In this case the issue lies in assigning weights allotted to a contractor. The bridge method would reduce redundancy by making sure these weights do not exceed 1.

References:

Rowen, W., Song, I. Y., Medsker, C., & Ewen, E. (2001). An analysis of many-to-many relationships between fact and dimension tables in dimensional modeling. In *International Workshop on Design and Management of Data Warehouses (DMDW 2001), Interlaken Switzerland* (pp. 1-13).



X. Appendix (Fix Lab #3 Problems)

**Rationale for Final Schema Design:** The designing of the final schema design, we merged the Supplier in the Product itself to keep the track the whole product values as the same. The reason behind it, because the two divisions can sell the same product, but prices and supplier can be different, so its easy to keep intact to track down the information.