***The Product Company***

**~  Final Data Mart Development Report ~**

Team # 2

**Team Members**

Rishi Wadekar

Rushikesh Kulkarni

Vivek Panchal

Himanshu Nirmal

Date

12th December 2019

ISTE-724   Data Warehousing

~  Table of Contents ~

[I.  Data Mart Design Definition 2](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.30j0zll)

[1. Universe of Discourse](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.1fob9te)  2

[2. Information Package](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.3znysh7)  2

[II.  Dimensional Model](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.2et92p0)  8

[III.  Data Staging: ETL – Data Extract File Definitions](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.tyjcwt)  9

[IV.  Data Staging: ETL – Source-to-Target Mappings](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.3dy6vkm)  18

[V.  SQL Code – Tables & Constraints](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.1t3h5sf)  19

[VI.  Data Staging Activities - ETL](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.4d34og8)  25

[1. Data Cleansing](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.2s8eyo1)  25

[2. Data Transformation](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.17dp8vu)  30

[3. Table Population](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.3rdcrjn)  33

[VII.  End User Applications](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.26in1rg)  35

[1. SQL99 Queries](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.lnxbz9)  35

[2. A Views](https://docs.google.com/document/d/152UkwJ9DFWDLTsvojgfcoYvN6hgYPhSqZf5QxRTdgfw/edit#heading=h.35nkun2)  38

3.   Aggregated Data Marts …….……………………………………………....38

VIII. Handling Slowly Changing Dimensions (SCD)…….…………………………. 51

IX. N-M Implementation Option………………………………………………….…56

X Appendix (Fix Lab #3) ………………………………………………………….59

I. Data Mart Design Definition

1. Universe of Discourse

|  |
| --- |
| The universe of discourse for the given data mart is to manage the sales of The Product Company through its three divisions namely PEC, TPCE, TPCW to its customers.  The data mart also manages the historical sales, reports, overall profit for all the divisions with a view to optimize sales with keeping costs lower and maintaining a healthy relationship with suppliers. |

2. Information Package

Process Name:   Financial Performance of TPC and its divisions.

Grain: Daily sales for each customer, product and supplier.

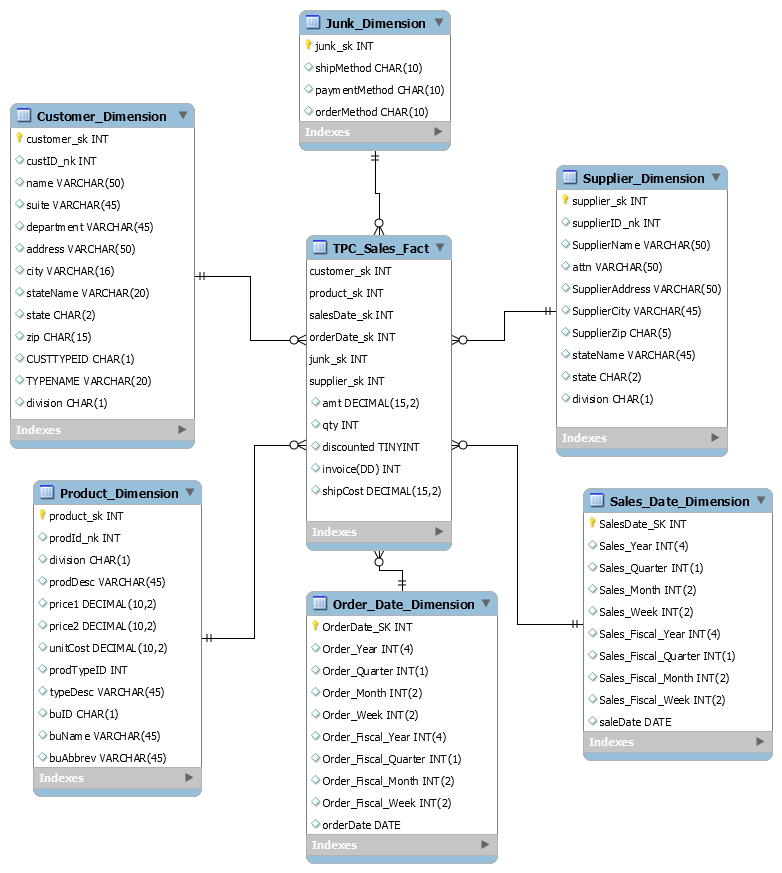
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Customer | Product | Supplier | Order\_Date | Sales\_Date | Junk |
| customer\_sk | product\_sk | supplier\_sk | OrderDate\_sk | SalesDate\_sk | junk\_sk |
| custID\_nk | prodId\_nk | supplierID\_nk | Order\_Year | Sales\_Year | shipMethod |
| name | division | SupplierName | Order\_Quarter | Sales\_Quarter | paymentMethod |
| suite | prodDesc | attn | Order\_Month | Sales\_Month | orderMethod |
| department | price1 | SupplierAddress | Order\_Week | Sales\_Week |  |
| address | price2 | SupplierCity | Order\_Fiscal\_Year | Sales\_Fiscal\_Year |  |
| city | unitCost | SupplierZip | Order\_Fiscal\_Quarter | Sales\_Fiscal\_Quarter |  |
| stateName | prodTypeID | stateName | Order\_Fiscal\_Month | Sales\_Fiscal\_Month |  |
| state | typeDesc | state | Order\_Fiscal\_Week | Sales\_Fiscal\_Week |  |
| zip | buID | division | orderDate | saleDate |  |
| CUSTTYPEID | buName |  |  |  |  |
| TYPENAME | buAbbrev |  |  |  |  |
| division |  |  |  |  |  |

Facts: amt, qty, discounted, shipCost

3. Entity Definitions

|  |  |
| --- | --- |
| **Entity** | **Entity Definition** (*genus differentia*) |
| **Customer\_Dimension** | This entity describes customers of Product company across all divisions i.e. PEC, TPCE, TPCW |
|  | 1.  customer\_sk - Surrogate key for customer dimension |
|  | 2.  custID\_nk - Natural key for customer dimension |
|  | 3. name - Name of customer |
|  | 4. suite - Suite number of customer’s address |
|  | 5. department - Department number of customer’s address |
|  | 6. address - Address of the customer |
|  | 7. city - City name where customer resides |
|  | 8. stateName - State name where customer resides |
|  | 9. state - State code where customer resides |
|  | 10. zip- Zip code of customer’s address |
|  | 11. CUSTTYPEID - Code for customer type |
|  | 12. TYPENAME - Name of customer type |
|  | 13. division - Numeric Division where customer belongs (PEC,TPCE,TPCW) |
|  |  |
| **Product\_Dimension** | This entity describes products of Product company across all divisions i.e PEC, TPCE,TPCW |
|  | 1. product\_sk - Surrogate key for product dimension |
|  | 2. prodId\_nk - Natural key for product dimension |
|  | 3. division - Numeric Division where product belongs (PEC,TPCE,TPCW) |
|  | 4. prodDesc - Description about Product |
|  | 5. price1 - Price when discount = 0 |
|  | 6. price2 - Price when discount = 1 |
|  | 7. unitCost - Cost of individual unit |
|  | 8. prodTypeID - Id obtained from product type |
|  | 9. typeDesc - Description of product type |
|  | 10. buID - Business Unit Id |
|  | 11. buName - Name of business unit company |
|  | 12. buAbbrev - Abbreviation of business unit company |
|  |  |
| **Supplier\_Dimension** | This entity describes supplier of Product company across all divisions i.e. PEC, TPCE,TPCW |
|  | 1.  supplier\_sk - Surrogate key of Supplier Dimension |
|  | 2. supplierID\_nk - Natural key of Supplier Dimension |
|  | 3. SupplierName - Name of the supplier |
|  | 4. attn - Representative for that supplier |
|  | 5. SupplierAddress - Address of supplier |
|  | 6. SupplierCity - City name of supplier |
|  | 7. SupplierZip - Zip code of supplier address |
|  | 8. stateName - Suppliers state name |
|  | 9.  state - State code of supplier |
|  | 10. division - Numeric division of supplier (PEC, TPCE, TPCW) |
|  |  |
| **Order\_Date\_Dimension** | It is a role-playing dimension in our dimensional model. It represents order date across all divisions namely (PEC, TPCE, TPCW) |
|  | 1. OrderDate\_sk - Surrogate key of Order Date |
|  | 2. Order\_Year - Year of order |
|  | 3. Order\_Quarter - Quarter when order was made |
|  | 4. Order\_Month - Month of order |
|  | 5. Order\_Week - Week of order |
|  | 6. Order\_Fiscal\_Year - Fiscal year of order |
|  | 7. Order\_Fiscal\_Quarter - Fiscal quarter of order |
|  | 8. Order\_Fiscal\_Month - Fiscal month of order |
|  | 9. Order\_Fiscal\_Week - Fiscal week of order |
|  |  |
| **Sales\_Date\_Dimension** | It is a role-playing dimension in our dimensional model. It represents sales date across all divisions namely (PEC, TPCE, TPCW) |
|  | 1. SalesDate\_sk - Surrogate key of Sales Date |
|  | 2. Sales\_Year - Year of sales |
|  | 3. Sales\_Quarter - Quarter when sales were made |
|  | 4. Sales\_Month - Month of sales |
|  | 5. Sales\_Week - Week of sales |
|  | 6. Sales\_Fiscal\_Year - Fiscal year of sales |
|  | 7. Sales\_Fiscal\_Quarter - Fiscal quarter of sales |
|  | 8. Sales\_Fiscal\_Month - Fiscal month of sales |
|  | 9. Sales\_Fiscal\_Week - Fiscal week of sales |
|  |  |
| **Junk\_Dimension** | This is a Junk Dimension as the nature of attributes is mostly flag like as they don’t belong to any dimension. It describes the type of shipping, payment and ordering method across all dimensions. |
|  | 1. junk\_sk - Surrogate key of junk dimension |
|  | 2. shipMethod - Represents different types of shipping methods |
|  | 3. paymentMethod - Represents different types of payment methods |
|  | 4. orderMethod - Represents different types of order method |
|  |  |
| **TPC\_Sales\_Fact** | This represents Fact table in dimensional model. |
|  | 1.customer\_sk - It is a surrogate key of customer dimension which acts as a foreign key and composite primary key in fact table |
|  | 2. product\_sk - It is a surrogate key of product dimension which acts as a foreign key and composite primary key in fact table |
|  | 3. supplier\_sk - It is a surrogate key of supplier dimension which acts as a foreign key and composite primary key in fact table |
|  | 4. orderDate\_sk - It is a surrogate key of order\_date dimension which acts as a foreign key and composite primary key in fact table |
|  | 5.salesDate\_sk - It is a surrogate key of csales\_date dimension which acts as a foreign key and composite primary key in fact table |
|  | 6. junk\_sk - It is a surrogate key of junk dimension which acts as a foreign key and composite primary key in fact table |
|  | 7. amt - Represents cost of product |
|  | 8. qty - Quantity of products that was ordered and sold at given time |
|  | 9. discounted - Represents whether product is discounted or not (0/1) |
|  | 10. invoice - It is a degenerate dimension in fact table |
|  | 11. shipCost - Represents the shipping costs involved |

II.  Dimensional Model

****

III.  Data Staging: ETL – Data Extract File Definitions

**Data Source - PEC**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **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\_type.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | CUSTTYPEID- String  TYPENAME-String |
| 3 | PECcustomer.csv | Fields separated by semicolon (;) | custID-Integer,  name-String,  address-String,  city-String,  state-String,  zip-Integer,  custType - 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- nteger |

**Data Source - TPCE**

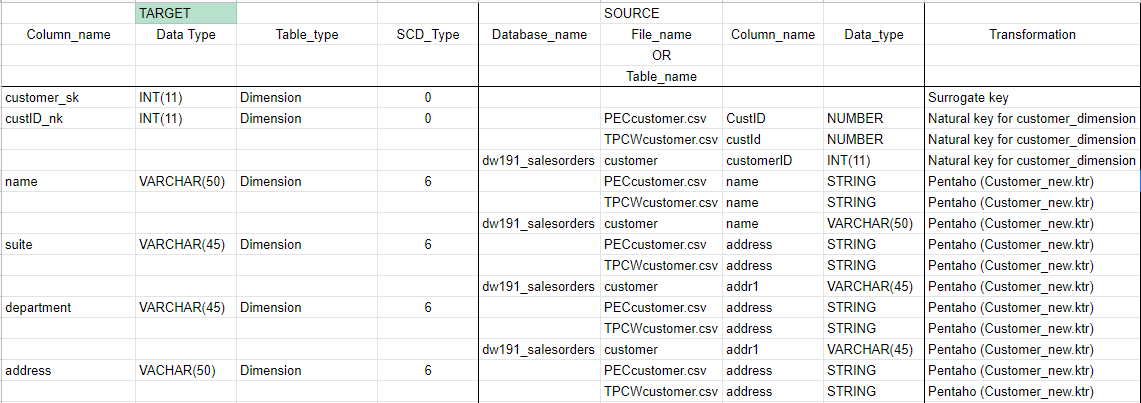
|  |  |  |  |
| --- | --- | --- | --- |
| # | **File Name** | **Format** | **Data Type** |
| 1 | business\_unit.csv | Fields enclosed in double quotes (“”) and separated by semicolon (;) | BUID – String,  NAME- String,  ABBREV-String |
| 2 | invoice.csv | Fields separated by comma (,) | InvoiceID-Integer,  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 |

**Data Source - TPCW**

|  |  |  |  |
| --- | --- | --- | --- |
| # | **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 |

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

The following is a snippet of the source-to-target mappings for our data mart. The image itself is a link to the Google Sheets where the entire mapping resides. Also, we have added the *mapping.xlsx* file along with this report.

***[](https://docs.google.com/spreadsheets/d/1g4DiWPJU3ENVbcWxeiD9M0FUdNviNNGYwGLr-qG0Sxk/edit?usp=sharing)***

V.  SQL Code – Tables & Constraints

**Creating `salesorders\_02\_2191` database**

**Query:**

CREATE DATABASE /\*!32312 IF NOT EXISTS\*/ `salesorders\_02\_2191` /\*!40100 DEFAULT CHARACTER SET utf8 \*/;

USE `salesorders\_02\_2191`;

**Customer Dimension**

**Query**:

DROP TABLE IF EXISTS `customer\_dimension`;

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

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `customer\_dimension` (

  `customer\_sk` int(11) NOT NULL,

  `custID\_nk` int(11) DEFAULT NULL,

  `name` varchar(50) DEFAULT NULL,

  `suite` varchar(45) DEFAULT NULL,

  `department` varchar(45) DEFAULT NULL,

  `address` varchar(50) DEFAULT NULL,

  `city` varchar(16) DEFAULT NULL,

  `stateName` varchar(20) DEFAULT NULL,

  `state` char(2) DEFAULT NULL,

  `zip` char(15) DEFAULT NULL,

  `CUSTTYPEID` char(1) DEFAULT NULL,

  `TYPENAME` varchar(20) DEFAULT NULL,

  `division` char(1) DEFAULT NULL,

  PRIMARY KEY (`customer\_sk`)

) ;

**Product Dimension**

**Query:**

DROP TABLE IF EXISTS `product\_dimension`;

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

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `product\_dimension` (

  `product\_sk` int(11) NOT NULL,

  `prodId\_nk` int(11) DEFAULT NULL,

  `division` char(1) DEFAULT NULL,

  `prodDesc` varchar(45) DEFAULT NULL,

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

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

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

  `prodTypeID` int(11) DEFAULT NULL,

  `typeDesc` varchar(45) DEFAULT NULL,

  `buID` char(1) DEFAULT NULL,

  `buName` varchar(45) DEFAULT NULL,

  `buAbbrev` varchar(45) DEFAULT NULL,

  PRIMARY KEY (`product\_sk`)

);

**Supplier Dimension**

**Query:**

DROP TABLE IF EXISTS `supplier\_dimension`;

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

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `supplier\_dimension` (

  `supplier\_sk` int(11) NOT NULL,

  `supplierID\_nk` int(11) DEFAULT NULL,

  `SupplierName` varchar(50) DEFAULT NULL,

  `attn` varchar(50) DEFAULT NULL,

  `SupplierAddress` varchar(50) DEFAULT NULL,

  `SupplierCity` varchar(45) DEFAULT NULL,

  `SupplierZip` char(5) DEFAULT NULL,

  `stateName` varchar(45) DEFAULT NULL,

  `state` char(2) DEFAULT NULL,

  `division` char(1) DEFAULT NULL,

  PRIMARY KEY (`supplier\_sk`)

);

**Order Date Dimension**

**Query:**

DROP TABLE IF EXISTS `order\_date\_dimension`;

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

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `order\_date\_dimension` (

  `OrderDate\_SK` int(11) NOT NULL,

  `Order\_Year` int(4) DEFAULT NULL,

  `Order\_Quarter` int(1) DEFAULT NULL,

  `Order\_Month` int(2) DEFAULT NULL,

  `Order\_Week` int(2) DEFAULT NULL,

  `Order\_Fiscal\_Year` int(4) DEFAULT NULL,

  `Order\_Fiscal\_Quarter` int(1) DEFAULT NULL,

  `Order\_Fiscal\_Month` int(2) DEFAULT NULL,

  `Order\_Fiscal\_Week` int(2) DEFAULT NULL,

  `orderDate` date DEFAULT NULL,

  PRIMARY KEY (`OrderDate\_SK`)

);

**Sales Date Dimension**

**Query:**

DROP TABLE IF EXISTS `sales\_date\_dimension`;

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

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `sales\_date\_dimension` (

  `SalesDate\_SK` int(11) NOT NULL,

  `Sales\_Year` int(4) DEFAULT NULL,

  `Sales\_Quarter` int(1) DEFAULT NULL,

  `Sales\_Month` int(2) DEFAULT NULL,

  `Sales\_Week` int(2) DEFAULT NULL,

  `Sales\_Fiscal\_Year` int(4) DEFAULT NULL,

  `Sales\_Fiscal\_Quarter` int(1) DEFAULT NULL,

  `Sales\_Fiscal\_Month` int(2) DEFAULT NULL,

  `Sales\_Fiscal\_Week` int(2) DEFAULT NULL,

  `saleDate` date DEFAULT NULL,

  PRIMARY KEY (`SalesDate\_SK`)

);

**Junk Dimension**

**Query:**

DROP TABLE IF EXISTS `junk\_dimension`;

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

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `junk\_dimension` (

  `junk\_sk` int(11) NOT NULL,

  `shipMethod` char(10) DEFAULT NULL,

  `paymentMethod` char(10) DEFAULT NULL,

  `orderMethod` char(10) DEFAULT NULL,

  PRIMARY KEY (`junk\_sk`));

**TPC\_Sales\_Fact**

**Query:**

DROP TABLE IF EXISTS `tpc\_sales\_fact`;

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

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `tpc\_sales\_fact` (

  `customer\_sk` int(11) NOT NULL,

  `product\_sk` int(11) NOT NULL,

  `salesDate\_sk` int(11) NOT NULL,

  `orderDate\_sk` int(11) NOT NULL,

  `junk\_sk` int(11) NOT NULL,

  `supplier\_sk` int(11) NOT NULL,

  `amt` decimal(15,2) DEFAULT NULL,

  `qty` int(11) DEFAULT NULL,

  `discounted` tinyint(4) DEFAULT NULL,

  `invoice(DD)` int(11) DEFAULT NULL,

  `shipCost` decimal(15,2) DEFAULT NULL,

  PRIMARY KEY (`customer\_sk`,`product\_sk`,`salesDate\_sk`,`orderDate\_sk`,`junk\_sk`,`supplier\_sk`),

  KEY `product\_fk\_idx` (`product\_sk`),

  KEY `salesDate\_fk\_idx` (`salesDate\_sk`),

  KEY `orderDate\_fk\_idx` (`orderDate\_sk`),

  KEY `junk\_fk\_idx` (`junk\_sk`),

  KEY `supplier\_fk\_idx` (`supplier\_sk`),

  CONSTRAINT `customer\_fk1` FOREIGN KEY (`customer\_sk`) REFERENCES `customer\_dimension` (`customer\_sk`) ON DELETE NO ACTION ON UPDATE NO ACTION,

  CONSTRAINT `junk\_fk1` FOREIGN KEY (`junk\_sk`) REFERENCES `junk\_dimension` (`junk\_sk`) ON DELETE NO ACTION ON UPDATE NO ACTION,

  CONSTRAINT `orderDate\_fk1` FOREIGN KEY (`orderDate\_sk`) REFERENCES `order\_date\_dimension` (`OrderDate\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

  CONSTRAINT `product\_fk1` FOREIGN KEY (`product\_sk`) REFERENCES `product\_dimension` (`product\_sk`) ON DELETE NO ACTION ON UPDATE NO ACTION,

  CONSTRAINT `salesDate\_fk1` FOREIGN KEY (`salesDate\_sk`) REFERENCES `sales\_date\_dimension` (`SalesDate\_SK`) ON DELETE NO ACTION ON UPDATE NO ACTION,

  CONSTRAINT `supplier\_fk1` FOREIGN KEY (`supplier\_sk`) REFERENCES `supplier\_dimension` (`supplier\_sk`) ON DELETE NO ACTION ON UPDATE NO ACTION

);

VI.  Data Staging Activities - ETL

1. Data Cleansing

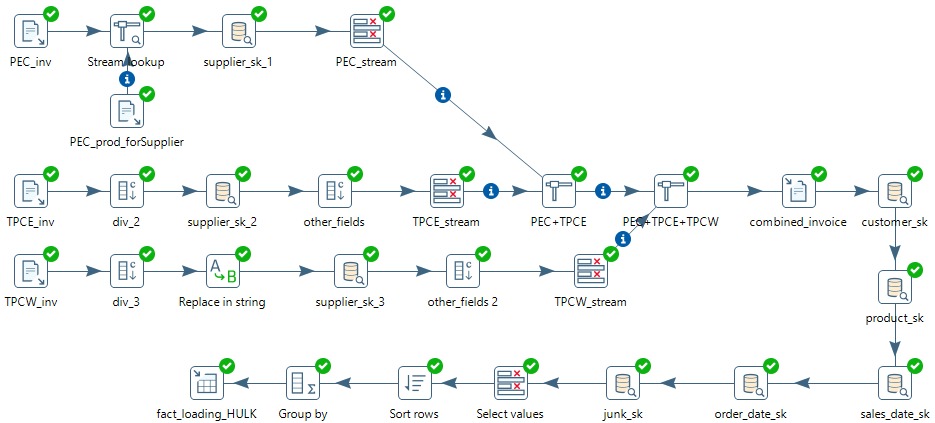
|  |  |  |  |
| --- | --- | --- | --- |
| **DM Table** | **Attribute** | **Problem** | **Resolution Strategy** (attach code) |
| **Customer\_Dimension** | | | |
| **PEC** | | | |
| PECcustomer.csv | address,custtype | Commas, periods, quotes,incorrect spellings in address  and custtype field | Used “Replace in string” to fix it. |
| PECcustomer\_type.csv | All attributes,  TYPENAME | 1.) Unwanted quotes in data.  2.) Changed Gov in TYPENAME | 1.) Used “Replace in String” to rectify it.  2.) To maintain uniformity changed it to Govt |
|  | state | Full names of states | Mapped state name to state code to make it user friendly. As, state codes might make it difficult to understand. |
| **TPCE** | | | |
| TPCECustomer | addr1, addr2 | Dept#. and Suite were present for only a few of the addresses | Addition of two new columns were done in this case using modified javascript. |
|  | names | names- Acronyms were present in the customer name field such as Inc/Inc./Co./Corp. | Used “Replace in String” to remove unnecessary values and to replace acronyms with appropriate value. |
|  | state, typeName | Only state acronyms were present instead of name of state  CA - California  typeName was State Local/Gov | Used a separate csv state\_data.csv to create a column named stateName.  Also used “Replace in string” to change typeName  Then merged both the files on custtypeid. |
| **TPCW** | | | |
| TPCWcustomer.csv | All the attributes | Unwanted commas, quotes and periods in every field name | Removed using “Replace in String” filter |
|  | zip | Zip Codes in 4 digits | Used “Modified Javascript” filter to fix this inconsistency and made them into 5 digits. |
|  | address | Dept and Suite Numbers included in address | Used “Modified Javascript” to fix it. |
|  | state | Inconsistent values in state field | Keeping user friendly approach in mind we created mapping through which state code and state name were rectified to maintain uniformity. |
| **Product\_Dimension** | | | |
| **PEC** | | | |
| PECproduct.csv | productDescription | spelling errors | Used “Replace in string” to fix it. |
|  | ProdId, productDescription,  productTypeId | Unwanted quotes in values | Used “Replace in string” to fix it. |
|  | unitCost | Unit costs are calculated using manufacturing cost | The calculation done to obtain unitPrice was done by using Manufacturing cost and Invoice data. Join operation is performed on manufacturing cost and Invoice.For the join we used the salesDate and quantity sold for every product during a particular month and year i.e(mm/yyyy).  unitPrice(product)=manufacturingCost(mm/yyyy)/totalQty(mm/yyyy) |
| **TPCE** | | | |
| TPCEProduct | description | Acronyms present in the description such as Equip. | Used “Replace in String” to replace acronyms with appropriate value. |
| **TPCW** | | | |
| TPCWproduct.csv | City,state | City and State values were combined. | Used field splitter functionality of Pentaho to fix these |
|  | All the attributes | Unwanted quotes in every field name | Used “Replace in String” filter to remove the quotes. |
|  | ProductId | Duplicate values for same product id | Used “Unique Rows” filter to fix these issues. |
|  | SupplierName | Inconsistencies in values such as  St. - Street  Ag. - Agency  Ave – Avenue  Co. – Company  Inc./Inc – Incorporated  Equip - Equipment | Used “Replace in String” filter to deal with such inconsistencies |
| TPCWproduct\_type.csv | All attributes | Unwanted quotes in fields | Used “Replace in String” to fix it. |
| **Supplier\_Dimension** | | | |
| **TPCE** | | | |
| TPCESupplier | name    state | Acronyms were present in the supplier name field such as Inc/Inc./Mo./Ag  Only state acronyms were present instead of the name of the state.  CA - California | Used “Replace in String” to replace acronyms with appropriate value  Used a separate csv state\_data.csv to create a column named stateName |
| **Order\_Date\_Dimension** | | | |
| **PEC** | | | |
| PECinvoice.csv | orderDate | Incorrect and untidy dates | Used “Select values” and “Modified Javascript” filter to correct it. |
| **Sales\_Date\_Dimension** | | | |
| **PEC** | | | |
| PECinvoice.csv | saleDate | Incorrect and untidy dates | Used “Select values” and “Modified Javascript” filter to correct it. |
| **TPCE** | | | |
| TPCEinvoicehdr | saleDate | Date is in dd/MM/yyyy format. | Using “Select Values” changed the Date to MM/dd/yyyy format |
| **TPCW** | | | |
| TPCWinvoice.csv | salesDate | Inconsistencies in date format as also there were few years with values in single digit | Used “Modified Javascript” Filter and wrote a script where data consistency was obtained |
| **Junk\_Dimension** | | | |
| **PEC** | | | |
| PECinvoice.csv | shipMethod | Incorrect spellings | Used “Replace in String” filter to correct those mistakes. |

2. Data Transformation

|  |  |
| --- | --- |
| **DM Table** | **Image Creation Process** (attach code) |
| Customer | 1) Select input from different sources i.e. PEC, TPCE & TPCW which were cleaned individually during the extraction processes.  2) To identify the individual customer according to the Division we added a constant field named “division” with values 1 = PEC, 2 = TPCE, 3 = TPCW.  3) In the PEC division we will filter out the TPCE & TPCW as the internal division sales are not considered according to Business Scenario from Lab02.  4) All the records are then sorted by name and merged as PEC and TPCE and then TPCW.  5) After merging all the sources, we will check for unique values in the records.  6)  We will complete the dimension by adding the surrogate keys and null records.  7)  Null records will be added using the MySQL script. (“null\_records.sql”)  **https://lh6.googleusercontent.com/1N-T9eQF9ou5bwloa93if9DcadkgsMUvNPYBV7ill6jayWYCe-o3QbKK8hMj29OrB3hluFahq_2L-pORa0Sw0-rNgwg8sxncXF2P63rZT3C8ZmQB19TMTk9KXh_o0kv0MqaKNcD-**  [Refer file: **Customer\_Dimension.ktr**] |
|
|
|
|
|
| Product | 1)We get the clean files related to product from the extraction phase for TPCE, TPCW, PEC.  2)” Add constants” field is added with three files to differentiate TPCW, TPCE & PEC  (value 1 = PEC, 2 = TPCE, 3 = TPCW.).  3) Merging TPCEproduct with PECproduct using “Append streams” feature of Pentaho and later joining these with TPCWproduct.  4)We trimmed spaces from the productDesc field so as to maintain its uniformity.  5)Finally, we have our Product Dimension ready  6)To finish our product dimension we add surrogate keys and null records.  [Refer file: **dimension\_loading.ktr**] |
|
|
| Supplier | 1)Select the base files from the extraction phase that were cleaned i.e. TPCW, TPCE  2)We merge both files and check for any duplicates as we need only unique values.  3)After creating the dimension we will add the surrogate keys.  4)The “null” records will be added using MySQL script.  Check file “null\_records.sql”.  [Refer file: **Supplier\_Dimension.ktr]** |
| Order\_Date | 1)We segregate orderDate records from PECproduct into  Order\_Day, Order\_Month, Order\_Year.  2)Applying “Unique rows” filter to remove duplicates.  3)Adding two fields Order\_Quarter, Order\_Week using Calculator feature in Pentaho.  4)Fiscal\_Year,Fiscal\_Quarter,Fiscal\_Month,Fiscal\_Week fields are derived using Javascript code and added in the table.  5)Lastly to complete dimension surrogate keys and null records are added as mentioned in steps above.  [Refer file: **dimension\_loading.ktr**] |
| Sales\_Date | 1)We segregate the “salesDate” record into saleDay, saleMonth, saleYear this is done using “Calculator” option in Pentaho.  2)We repeat these steps across TPCE, TPCW, PEC.  3) Then by using “Append streams” feature of Pentaho and first join TPCWSales\_Date and TPCESales\_Date.  4)The result from the above step is then joined with PECSales\_Date.  5)Applying “Unique rows” filter to remove duplicates.  6)Adding two fields namely Sales\_Quarter, Sales\_Week which is calculated using “Calculator” feature of Pentaho.  7)Later, we added fields Fiscal\_Year, Fiscal\_Quarter, Fiscal\_Month, Fiscal\_Week which was calculated by writing Javascript code for which we made use of “Modified Javascript” feature.  8) Sort rows to maintain convenience and by following these steps we derived our Sales\_Date dimension.  9)To complete the dimension surrogate keys and null records are added as mentioned above.  [Refer file: **Sales\_Date.ktr**] |
| Junk | 1)Use the PECinvoice csv file as the input for this step.  2)Extract shipMethod, orderMethod and paymentMethod from the PEC invoice.  3)Then we will use “Sort Rows” to check for duplicate values.  4)Using the “Unique Rows” we will remove the duplicate files.  5) Adding surrogate keys and null record to complete the dimension as mentioned above.  [Refer file: **dimension\_loading.ktr**] |

3. Table Population

|  |  |
| --- | --- |
| **DM Table** | **Table Population Process** (attach code) |
| Customer | https://lh3.googleusercontent.com/k_9DHlwy9Ji8FmF6kgL8mwN3M0lNxS5VjDv7csrFulH8FlhRu0P5gPlYE-9lFalu2O8bIS-OANLy-iaNsicKGPq44rydqDaVZTuvzIWWFDLsOxKaBfEDfd7NPvry_wQxwyQBVw95  Here , we have added the Surrogate Keys for individual dimension CSVs as created above. And loaded the tables directly into the database. (sk = surrogate key) |
| Product |
| Supplier |
| Order\_Date |
| Sales\_Date |
| Junk |
|  |
|
|



1) Invoice details from TPCE, TPCW & PEC are gathered. We performed a join on TPCW\_Invoice and TPCW\_product using (productId) by which we got “supplierId”.

2) Dimensions (Supplier, Product, Customer, Order\_Date, Sales\_Date, Junk) are added with a lookup to get the corresponding surrogate keys.

3) The lookup step enables us to look up data using information coming from other steps in the transformation. The data coming from the Source step is first read into memory and is then used to look up data from the mainstream.

4) Removal of redundant fields is done (Fields not in the fact table). Also, fields with

 null value is replaced by 0 (null record from dimensions). We sorted rows to perform

grouping.

5) **GROUP BY** was performed because there were fields with same SK which shows

error in database when populated so those fields were aggregated into one.

These were the steps were performed to populate Fact\_Table in the database

VII.  End User Applications

1. Queries

|  |
| --- |
| **User Question/Reporting Need** |
| 1.The number of orders that are not shipped within 10 days of order from PEC |
| **SQL Code** |
| SELECT pd.division, COUNT(\*) 'No. of Orders from PEC not shipped within 10 days'  FROM tpc\_sales\_fact tsf  JOIN product\_dimension pd USING(product\_sk)  JOIN sales\_date\_dimension sd USING(salesdate\_sk)  JOIN order\_date\_dimension od USING(orderDate\_sk)  WHERE  pd.division = 1  AND  DATEDIFF(sd.saleDate,orderDate) >= 10;  **Result**:  https://lh5.googleusercontent.com/E3uaMeZEqAQQo_zSqvCosRN6ZfF7w7UOQ9mSWOC62gOHTCiV0w8kWznP2xBuqvKICwMf-hUE_Ap1K9Axs6EdFxdcNW7JgXEEdMdusWwL87b9mWXQa6SImGSWg0ZhjDn4VmSTRvQD |
| **Supporting Index(es)** |
| division(product\_dimension) |

|  |
| --- |
| **User Question/Reporting Need** |
| 2. The most frequent method of ordering a product from PEC. |
| **SQL Code** |
| SELECT jk.orderMethod, count(\*) 'Frequency'  FROM tpc\_sales\_fact tsf JOIN junk\_dimension jk USING (junk\_sk)  JOIN product\_dimension p USING (product\_sk)  WHERE p.division = 1  GROUP BY jk.orderMethod  ORDER BY Frequency DESC  LIMIT 1;  **Result**:  https://lh4.googleusercontent.com/ihcieNFEAYty-ZOGZhqneFlLjCZz-timPS5k3wEeFs90jCjsQTgdmaNgDw_qfo6sEHkfgmw2Ja0ks-Apf9eyw8JAOEEhmTGBVwdpDuZYeEbFSWhzMeMzLLYLmE9xBup5PIMiZNj5 |
| **Supporting Index(es)** |
| orderMethod(junk\_dimension) |

|  |
| --- |
| **User Question/Reporting Need** |
| 3. The average time in days needed to fulfill an order from PEC. |
| **SQL Code** |
| SELECT Division, AVG(a.diff) 'Average Time'  FROM  (SELECT pd.division 'Division', DATEDIFF(sd.saleDate,od.orderDate) 'diff'  FROM tpc\_sales\_fact tsf  JOIN product\_dimension pd USING(product\_sk)  JOIN sales\_date\_dimension sd USING(salesdate\_sk)  JOIN order\_date\_dimension od USING(orderDate\_sk)  WHERE pd.division = 1  GROUP BY pd.division) a;  **Result**:  https://lh3.googleusercontent.com/mNyMLELdzYTI-V4M6zkXqZpecib9T9uJCArWkcEKsmwnjF0MIMmF3w8mjNg6nsCPvgTQZf0j-IkfC3r3Nvb-AyF_AMMa_bvRC03vnxufJ1Yt-gq4xmMPvKXb_tlK9nKWDkd4Uvns |
| **Supporting Index(es)** |
| division(product\_dimension) |

|  |
| --- |
| **User Question/Reporting Need** |
| 4. The average cost of shipping for a particular product by different methods |
| **SQL Code** |
| SELECT a.Product, a.sm 'Shipping Method', a.asc1 'Average Shipping Cost'  FROM  (SELECT p.prodDesc 'Product', j.shipMethod 'sm', AVG(tsf.shipCost) 'asc1'  FROM tpc\_sales\_fact tsf  JOIN product\_dimension p USING(product\_sk)  JOIN junk\_dimension j USING(junk\_sk)  GROUP BY p.prodDesc, j.shipMethod) a  ORDER BY a.asc1 DESC  LIMIT 10;  **Result** :  https://lh3.googleusercontent.com/CQBZDcPnkMSyF_FqgNxLCIlSgc0kwkmfuZ5rKUWbGWKuUP1DXS2nHxRtO5wiDrefusoJdgTACebCGGhaNeapLXO1w_rUOgi_NRk1L0-K7fnMTzPX2z0wCeAo6DIcBgrY9ixGlzAl |
| **Supporting Index(es)** |
| prodDesc(product\_dimension), shipMethod(junk\_dimension) |

|  |
| --- |
| **User Question/Reporting Need** |
| 5. The percentage of invoices that are COD. |
| **SQL Code** |
| SELECT a.pm 'Payment Method', a.totalcod 'Total COD Invoices', b.total 'Total Invoices', format(100\*a.totalcod/b.total,2) "PCT"  FROM  (SELECT j.paymentMethod 'pm', COUNT(\*) 'totalcod'  FROM tpc\_sales\_fact tsf  JOIN junk\_dimension j USING(junk\_sk)  WHERE j.paymentMethod = "cod"  GROUP BY j.paymentMethod) a,  (SELECT COUNT(\*) 'total'  FROM tpc\_sales\_fact) b;  **Result**:  https://lh3.googleusercontent.com/BycrPFaGPpQyFk0iZ8lIguzKB87xuQI5N21f8s7UhaEFApIGPRTnCIFHnQGSePPHJ7jduTDxAuBbuJuikG8JpNSkZ_MHOx81y7Iq3CrhYO2d1_1Rt2HH87vQPthJzqJA5gAch5X9 |
| **Supporting Index(es)** |
| paymentMethod(junk\_dimension) |

2. A View

**We have created a view which reports sales, costs and gross profit (sales minus costs).**

**Query to create VIEW:**

CREATE VIEW gross\_profits\_saleYear AS

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

FROM

(SELECT s.Sales\_Year 'Year', SUM(tsf.amt) 'Sales', SUM(p.unitCost \* tsf.qty) 'Costs'

FROM tpc\_sales\_fact tsf JOIN product\_dimension p USING (product\_sk)

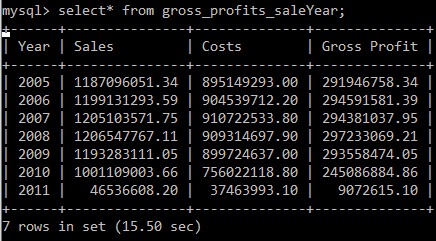
JOIN sales\_date\_dimension s USING(salesDate\_sk)

GROUP BY s.Sales\_Year) a;

**Query to fetch data from VIEW**:

Select \* from gross\_profits\_saleYear;

**Result:**

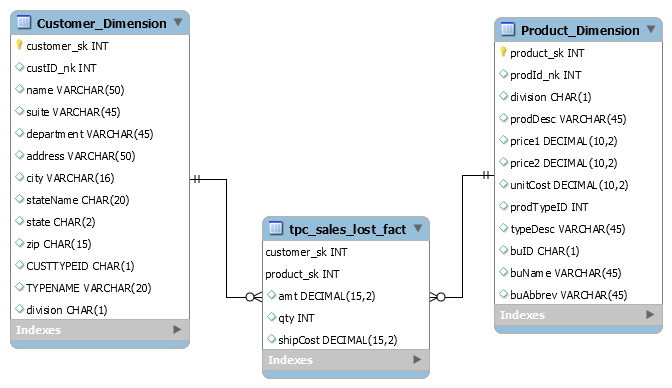


3. Aggregated Data Marts

We have implemented three aggregated data marts:

1. **Lost** **Aggregated Data Mart**

ERD:



Aggregation method: Lost Dimensions

Use Case:

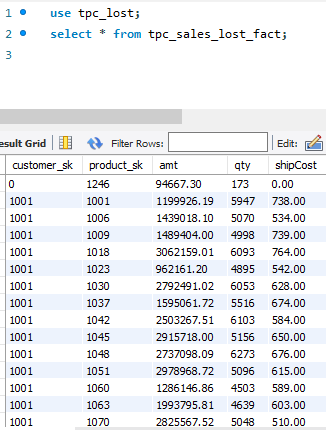
In the lost aggregated data mart, we collectively decided to only keep the product and customer dimensions. In this data mart, the user can focus the queries based only on the product and customer dimensions.

Creation of Tables:

We created the tables in MySQL Workbench and then used “Forward Engineer” to create the data mart.

File: TPC\_Lost.mwb

Following is an example of the fact table of the database:



Populating Lost Aggregated Data Mart:

For customer\_dimension:

INSERT INTO tpc\_lost.customer\_dimension(customer\_sk, custID\_nk,

name, suite, department, address, city, stateName, state,

zip, custtypeid, typename, division)

SELECT customer\_sk, custID\_nk, name, suite, department, address,

city, stateName, state, zip, custtypeid, typename, division

FROM salesorders\_02\_2191.customer\_dimension;

For product\_dimension:

INSERT INTO tpc\_lost.product\_dimension(product\_sk, prodId\_nk, division,

prodDesc, price1, price2, unitCost, prodTypeID, typeDesc, buID,

buName, buAbbrev)

SELECT product\_sk, prodId\_nk, division, prodDesc, price1, price2,

unitCost, prodTypeID, typeDesc, buID, buName, buAbbrev

FROM salesorders\_02\_2191.product\_dimension;

For tpc\_sales\_lost\_fact:

INSERT INTO tpc\_lost.tpc\_sales\_lost\_fact(customer\_sk, product\_sk, amt,

qty, shipCost)

SELECT customer\_sk, product\_sk, SUM(amt), SUM(qty), SUM(shipCost)

FROM salesorders\_02\_2191.tpc\_sales\_fact

GROUP BY customer\_sk, product\_sk;

User Queries:

i. Max amount spent by each customer, and the corresponding product type.

Query:

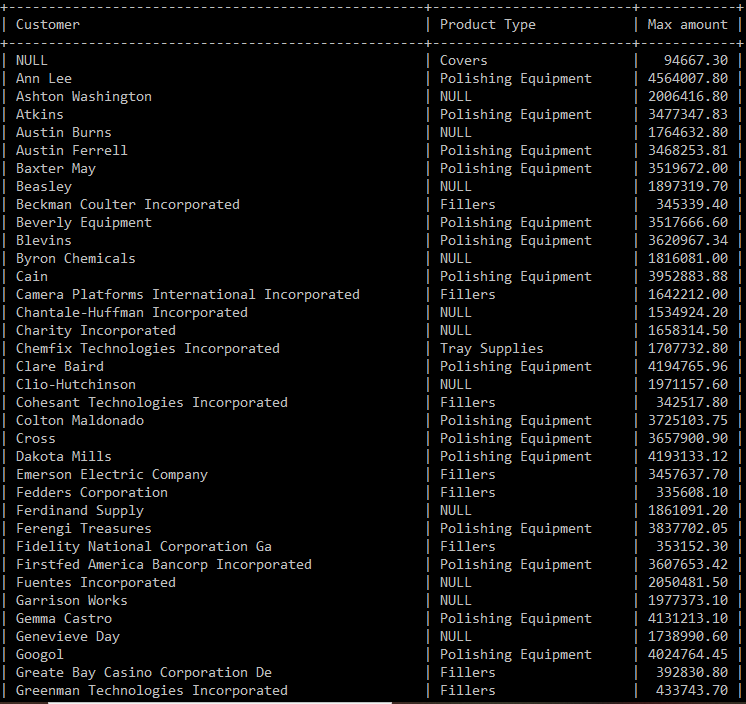
SELECT c.name 'Customer', p.typeDesc 'Product Type', MAX(f.amt) 'Max amount'

FROM tpc\_sales\_lost\_fact f

JOIN customer\_dimension c USING (customer\_sk)

JOIN product\_dimension p USING (product\_sk)

GROUP BY c.name;



ii. Business Units which sold more than 150000 products by state.

Query:

SELECT p.buName 'Business Unit', c.stateName 'State', SUM(f.qty) 'Quantity Sold'

FROM tpc\_sales\_lost\_fact f

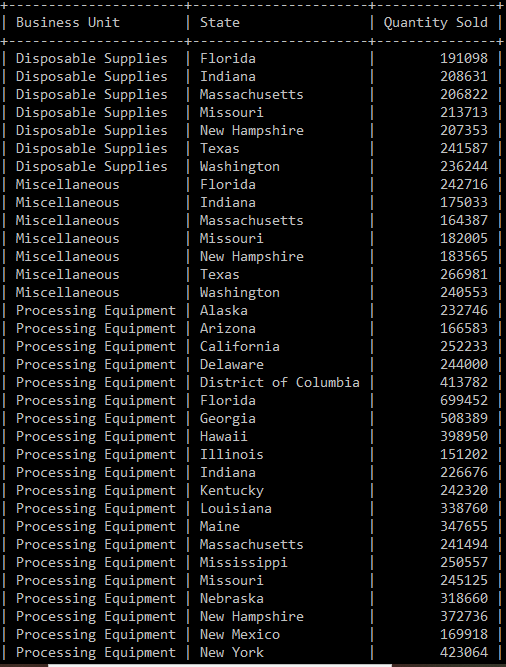
JOIN customer\_dimension c USING (customer\_sk)

JOIN product\_dimension p USING (product\_sk)

GROUP BY p.buName, c.stateName

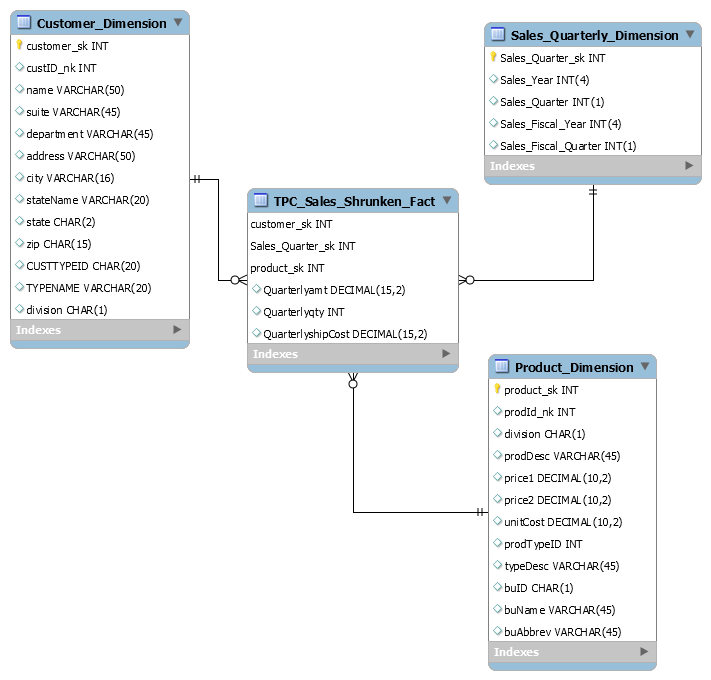
HAVING `Quantity Sold` > 150000

ORDER BY p.buName ASC, c.stateName ASC;



**2. Shrunken** **Aggregated Data Mart**

ERD:



Aggregation Method: Shrunken Dimension

Use Case:

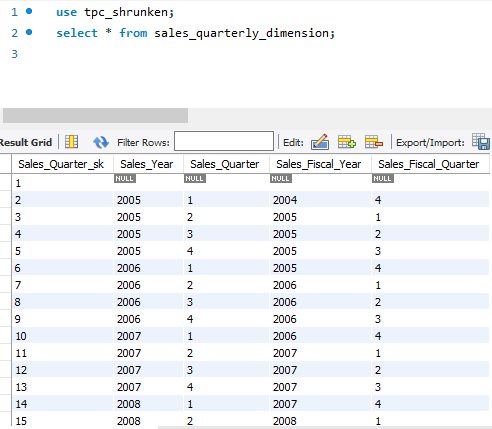
In the shrunken aggregated data mart, we shrink the dimensions to get a rolled-up form of the fact table. In this data mart, we shrunk the sale\_date\_dimension to take the grain from daily to quarterly. The user can use this data mart to run queries which focus on sale\_date\_Quarter.

Creation of Tables:

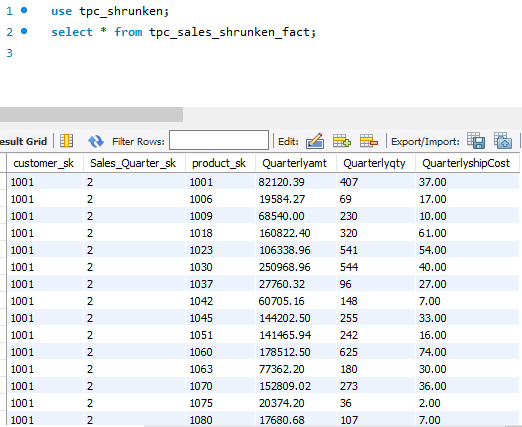
We created the tables in MySQL Workbench and then used “Forward Engineer” to create the data mart.

File: TPC\_Shrunken.mwb

Shrunken Dimension: Sales\_Quarter\_Dimension & Grain : Quarter



Following is an example of the fact table of the database:



Populating Shrunken Aggregated Data Mart:

**For customer\_dimension:**

INSERT INTO tpc\_shrunken.customer\_dimension(customer\_sk, custID\_nk, name,

suite, department, address, city, stateName, state, zip,

custtypeid, typename, division)

SELECT customer\_sk, custID\_nk, name, suite, department, address, city,

stateName, state, zip, custtypeid, typename, division

FROM salesorders\_02\_2191.customer\_dimension;

**For product\_dimension:**

INSERT INTO tpc\_shrunken.product\_dimension(product\_sk, prodId\_nk, division,

prodDesc, price1, price2, unitCost, prodTypeID, typeDesc, buID,

buName, buAbbrev)

SELECT product\_sk, prodId\_nk, division, prodDesc, price1, price2, unitCost,

prodTypeID, typeDesc, buID, buName, buAbbrev

FROM salesorders\_02\_2191.product\_dimension;

**For sales\_quarterly\_dimension:**

INSERT INTO tpc\_shrunken.sales\_quarterly\_dimension(sales\_year, sales\_quarter,

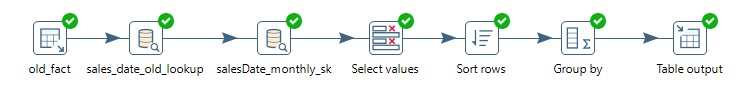
sales\_fiscal\_year, sales\_fiscal\_quarter)

SELECT sales\_year, sales\_quarter, sales\_fiscal\_year, sales\_fiscal\_quarter

FROM salesorders\_02\_2191.sales\_date\_dimension

GROUP BY sales\_year, sales\_quarter, sales\_fiscal\_year, sales\_fiscal\_quarter;

**For tpc\_sales\_shrunken:**

File: shrunken.ktr

User Queries:

i. Average ship cost per quarter per year.

Query:

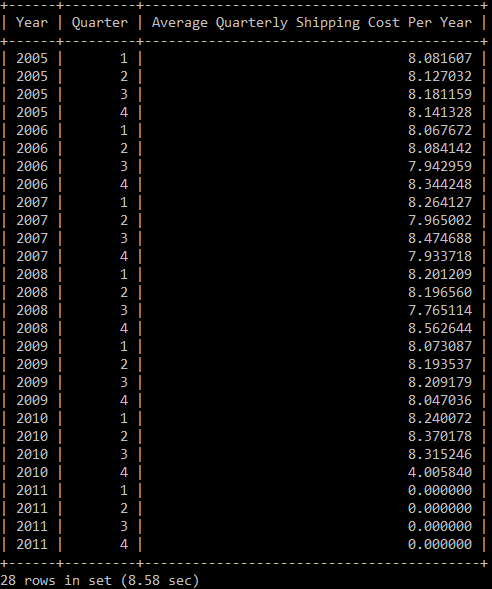
SELECT s.sales\_year 'Year', s.sales\_quarter 'Quarter',

AVG(f.QuarterlyshipCost) 'Average Quarterly Shipping Cost Per Year'

FROM tpc\_sales\_shrunken\_fact f

JOIN sales\_quarterly\_dimension s USING(sales\_quarter\_sk)

GROUP BY s.sales\_year, s.sales\_quarter;



ii. Quarterly amount for all years together.

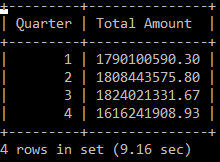
Query:

SELECT s.sales\_quarter 'Quarter', SUM(f.Quarterlyamt) 'Total Amount'

FROM tpc\_sales\_shrunken\_fact f

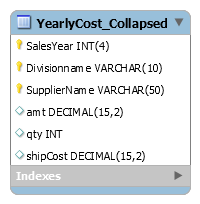
JOIN sales\_quarterly\_dimension s USING(sales\_quarter\_sk)

GROUP BY s.sales\_quarter;



**3. Collapsed** **Aggregated Data Mart**

ERD:

****

Aggregation method: Collapsed Dimension

Use Case:

In the collapsed aggregated data mart, we combine several dimensions into the fact table and remove the surrogate keys from it. Then we aggregate the data mart on the based on the attributes in fact table. Here, we have used SalesYear from Sales\_Date\_Dimension , division and SupplierName from the Supplier\_Dimension to gain knowledge regarding the transactions between TPC and its suppliers.

Creation of Tables:

We created the tables in MySQL Workbench and then used “Forward Engineer” to create the data mart.

File: TPC\_Collapsed.mwb

Populating Collapsed Aggregated Data Mart:

INSERT INTO tpc\_collapsed.yearlycost\_collapsed(salesyear,divisionname,suppliername,amt,qty,shipcost)

SELECT Sales\_Year , CASE WHEN division = 1 THEN "PEC"

WHEN division = 2 THEN "TPC East"

WHEN division = 3 THEN "TPC West"

END AS "DivisionName" , SupplierName, sum(amt) 'Yearly cost', sum(qty), sum(shipcost)

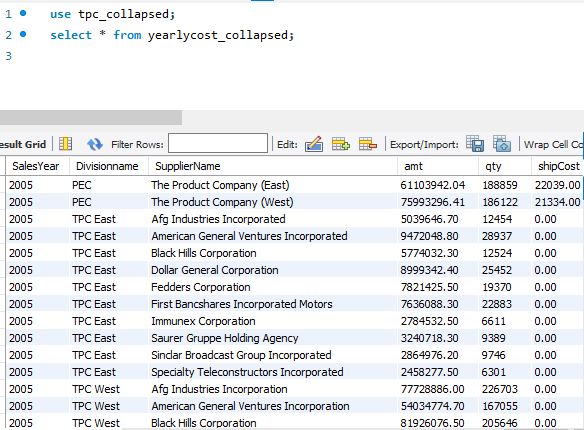
FROM salesorders\_02\_2191.tpc\_sales\_fact f JOIN salesorders\_02\_2191.sales\_date\_dimension sd

USING (salesDate\_sk) JOIN salesorders\_02\_2191.supplier\_dimension ss USING (supplier\_Sk)

WHERE division IS NOT NULL AND SupplierName IS NOT NULL

GROUP BY sd.sales\_year,ss.division,ss.suppliername ORDER BY 1,2,3;

Following is an example of the fact table of the database:



User Queries:

i. Top Supplier from individual divisions i.e. PEC, TPC East,TPC West

SELECT a.DivisionName, a.SupplierName, a.TotalCost, COUNT(b.TotalCost) 'Rank'

FROM (SELECT DivisionName,SupplierName, sum(amt) 'TotalCost'

FROM yearlycost\_collapsed y

GROUP BY y.DivisionName) a

JOIN

(SELECT DivisionName,SupplierName, sum(amt) 'TotalCost'

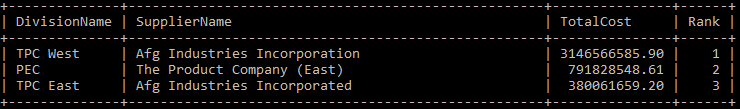
FROM yearlycost\_collapsed z

GROUP BY z.DivisionName) b

ON a.TotalCost <= b.TotalCost

GROUP BY a.DivisionName

ORDER BY 4;



ii. Side by side comparison of each year’s total transactions between supplier and PEC.

SELECT a.Divisionname, a.SalesYear, a.YearlyCost, b.SalesYear, b.YearlyCost

FROM (SELECT divisionName,SalesYear, sum(amt) AS 'YearlyCost'

FROM yearlycost\_collapsed y

WHERE divisionName = 'PEC'

GROUP BY y.SalesYear)a

JOIN

(SELECT divisionName,SalesYear, sum(amt) AS 'YearlyCost'

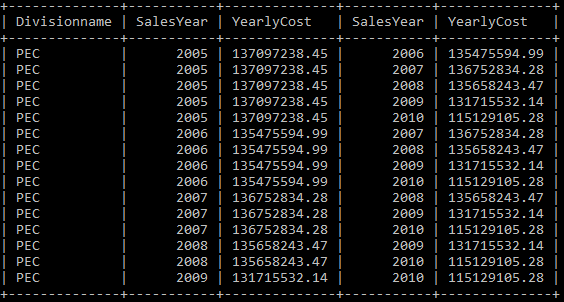
FROM yearlycost\_collapsed y

WHERE divisionName = 'PEC'

GROUP BY y.SalesYear)b

ON a.SalesYear < b.SalesYear

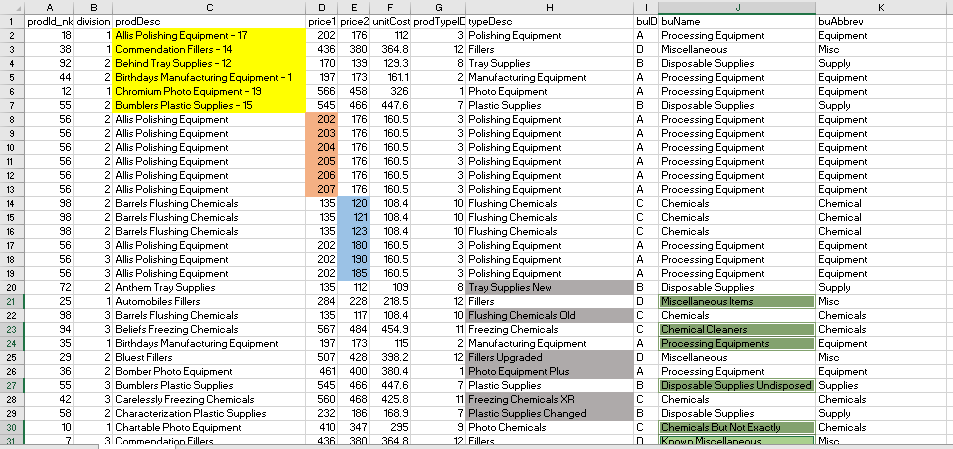
ORDER BY a.SalesYear,b.SalesYear;



VIII. Handling Slowly Changing Dimensions (SCD)

Source Data: newData.csv

Total rows: 30 (6 new rows for each SCD attribute)



Highlighted parts:

prodDesc (yellow): changes to implement SCD-1

price1 (peach): changes to implement SCD-2

price2 (blue): changes to implement SCD-2

typeDesc (grey): changes to implement SCD-1

buName (green): changes to implement SCD-1

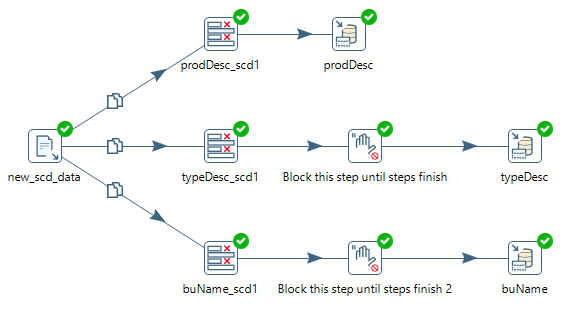
**SCD Type 1**:

Why?

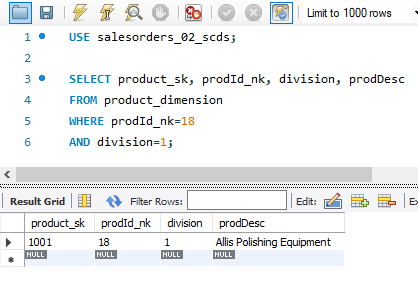
We implemented SCD Type-1 on the columns prodDesc, typeDesc and buName from the product\_dimension. We did so because the history of these attributes is not necessary to be maintained.

Implementation:

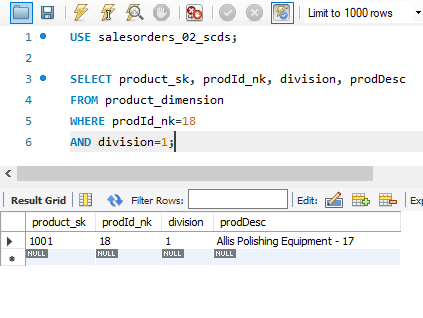
We implemented SCD-1 using Pentaho transformation (scd1.ktr). In the transformation, we used the step *Output → Insert / update* to implement SCD Type-1.



Original Record:



Updated Record:



Here, you can see that the record’s prodDesc field got updated after running the ktr file.

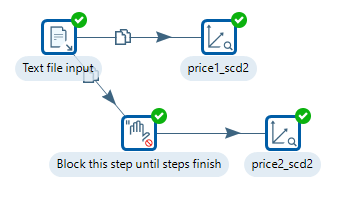
**SCD Type 2**:

Why?

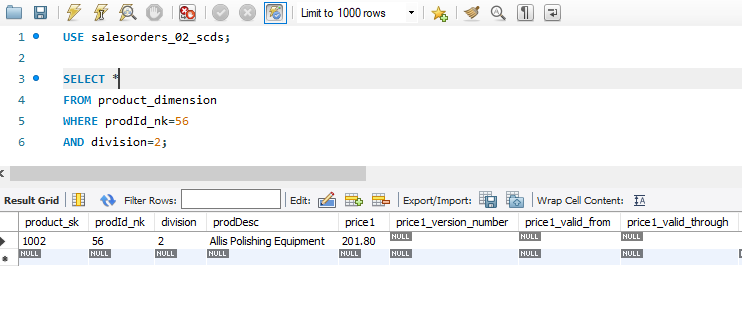
We implemented SCD Type-2 on the columns price1 and price2 from the product\_dimension. We did so because the history of these attributes is important and should be maintained.

Implementation:

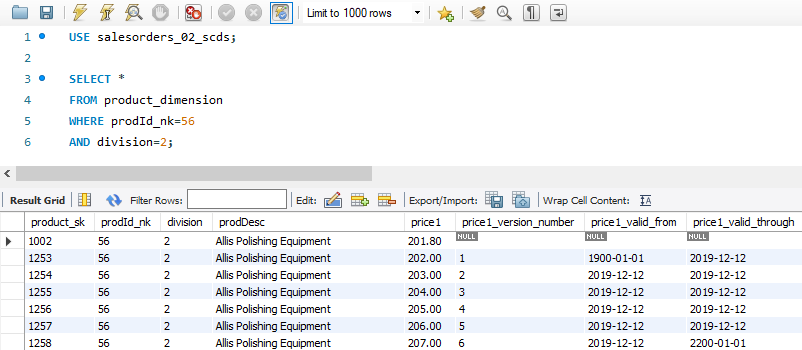
We implemented SCD-2 using Pentaho transformation (scd2.ktr). In the transformation, we used the step *Data Warehouse → Dimension lookup/ update* to implement SCD Type-2.



Original Record:



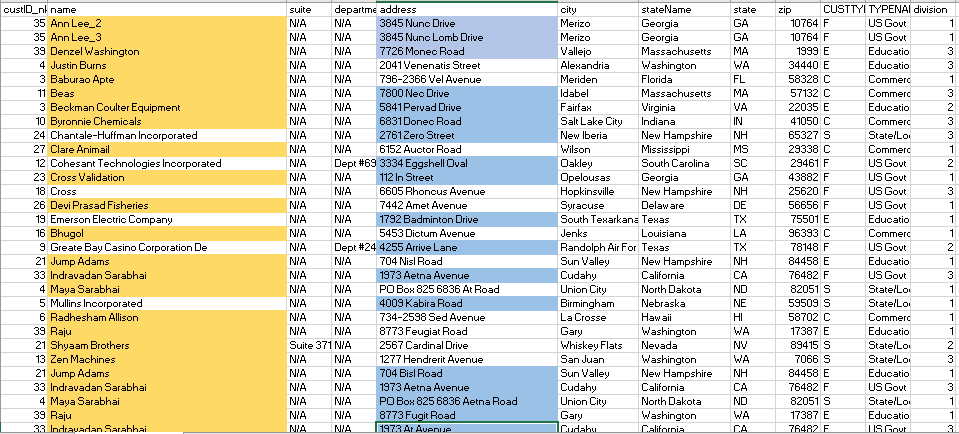
Updated Record:



**SCD Type 6**:

Source Data: newData\_scd6.csv

Total rows: 30

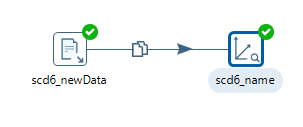


Why?

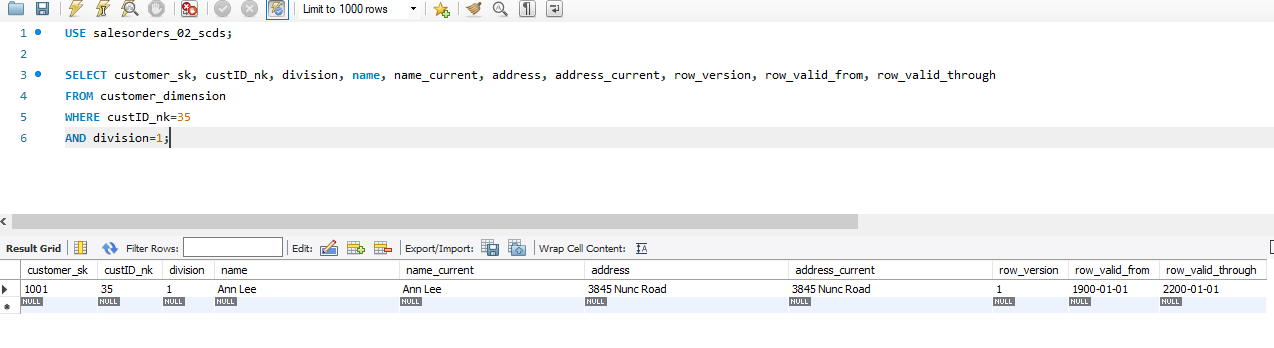
We implemented SCD Type-6 on the columns name and address from the customer\_dimension. We did so because the history of these attributes is important and should be maintained. Also, Type 6 SCD adds a current field which helps us to determine the current record and the date it is valid till.

Implementation:

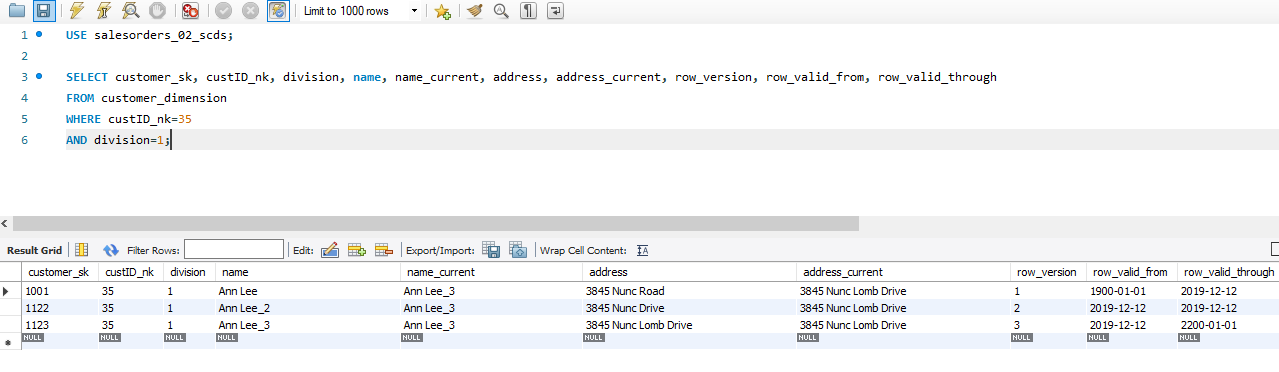
We implemented SCD-6 using Pentaho transformation (scd6.ktr). In the transformation, we used the step *Data Warehouse → Dimension lookup/ update* to implement SCD Type-6.



Original Record:



Updated Record:



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

There are times when we come across handling many to many relationships between a fact and dimension. This is somewhat violation of dimensional modeling rules. There are few methods to implement this multivalued relationship:

* **Method 1**

This is the most straightforward method to implement. We create a separate relationship table and add attribute ID’s to the fact table.

In the case given, if we want to measure the performance of the supplier, we should connect it with the shipping\_company dimension. So, using Method 1, we can create a relationship table (similar to an associative entity in rdbms) connecting the supplier and shipping\_company dimensions using their surrogate keys.

* **Method 2**

This method eliminates the many to many relationships and need for a distinct relationship table. We convert the many to many relationships into compound attribute relationship. It simply means that we treat one attribute as a child of other and have a compound key for lower level attribute.

For example, in the scenario given, using Method 2, we can have the supplier table to be a lookup table referencing the product dimension. So, the supplierId will be an attribute of the product\_dimension which looks up the supplier information from the supplier table.

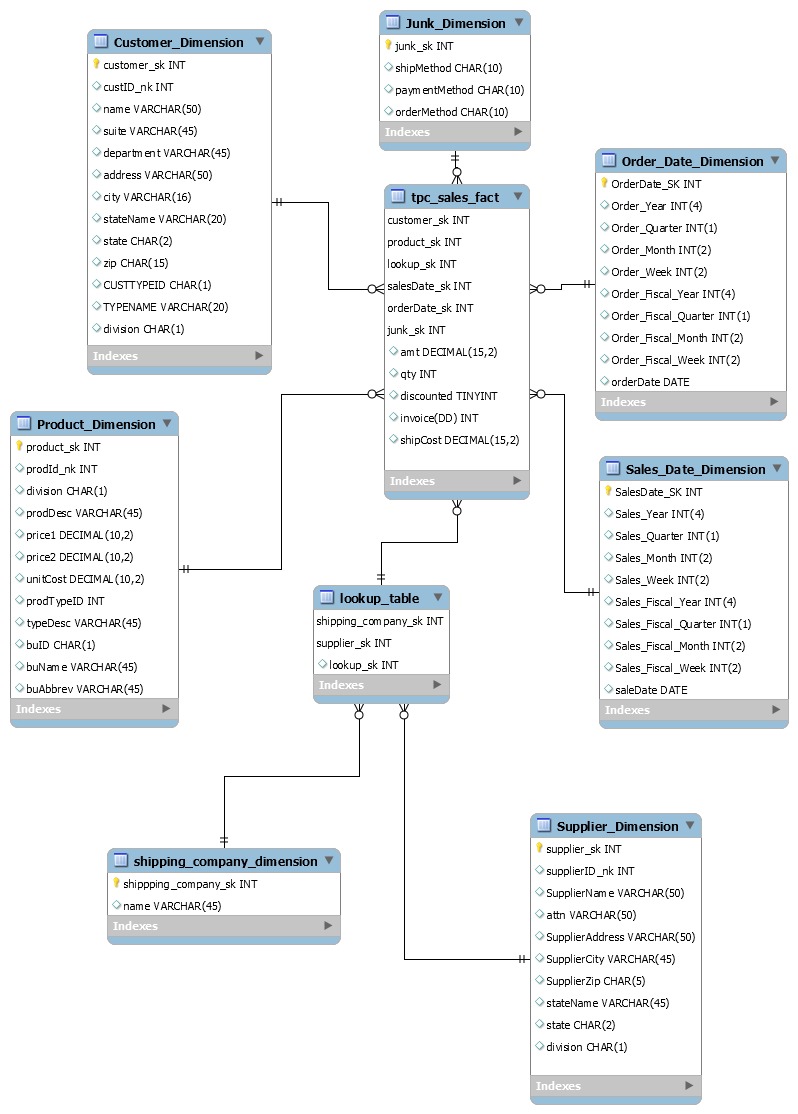
* **Method 3**

It is an adaptable solution and has many characteristics.

1. It simplifies the compound attribute relationship problem from method 2.
2. It requires only one of attribute columns in fact table rather than all the columns.
3. A new attribute is created which is lower in level than either of them. This attribute is essentially a concatenation of surrogate keys between its parent attributes.
4. Hence rather than including supplier\_sk and Shipping\_company\_sk in fact table we add this SKU attribute.

The thing which makes it different from method 1 is that we have a distinct relationship table from method 1 as an additional column but in this case, we have an SKU which extends relationship and combines in a single value. This can be later used in fact table.

The disadvantage of this method lies in its creation. If the current business model doesn’t use a similar structure it increases the complexity in ETL process.



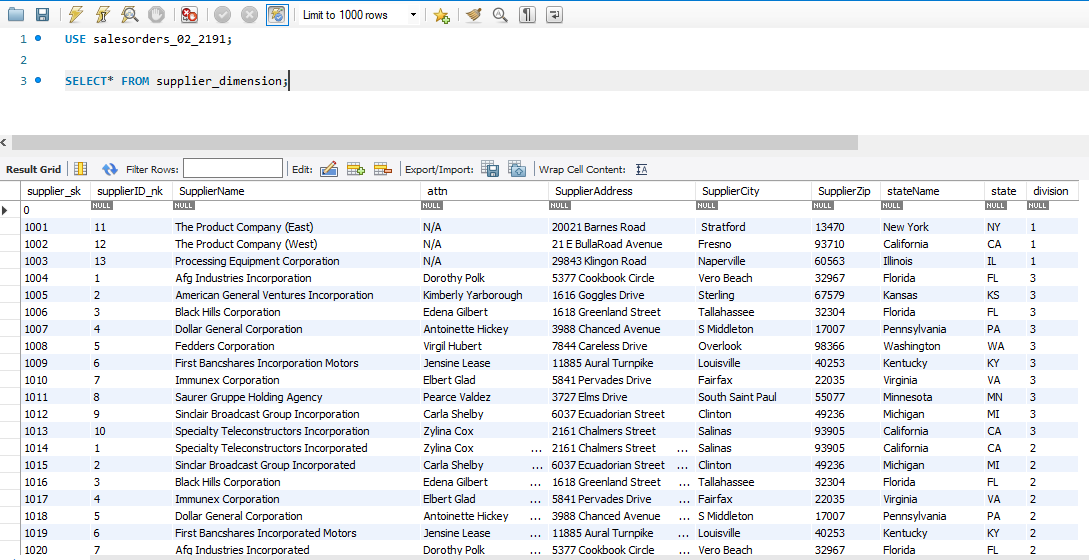
References:

Supporting many-to-many and joint child relationships. (n.d.). Retrieved from

<https://www2.microstrategy.com/producthelp/10.4/ProjectDesignGuide/WebHelp/Lang_1033/Content/ProjectDesign/Supporting_many_to_many_and_joint_child_relationsh.htm>

X. Appendix (Fix Lab #3 Problems)

Updated supplier\_dimension



Added null record to all dimensions.

Added PEC, TPCE and TPCW as suppliers.

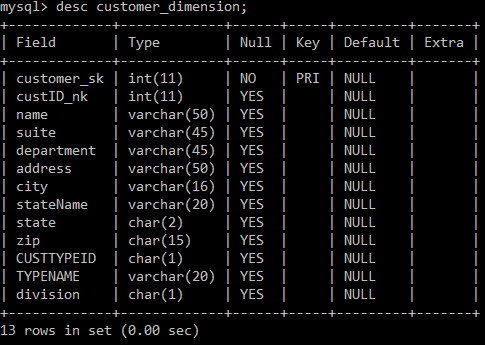
Added division attribute to differentiate between suppliers from different divisions.

Shifting column for TPCWInvoice(id = 45461)

https://lh6.googleusercontent.com/q-p8OPPVePRfzqdOEnJ7J7U8lykvosH3e6cz2w9SAWUbNuQ4Qg_FjUUKaUoRL1czgb0EzWM1O2A5nudLkYRFjJ9sSIyY-PrhzZIpodOmgP0qV3osoZ0t0sED518kBU5SIx3MdPPG

Updated TPCWinvoice where discounted was null for invoiceId=45461.

Customer Dimension (Split address fields for suite, dept,PO-Box)



Fact Table updated to include all records (~272k)

