*The Product Company*

~ Final Data Mart Development Report ~

Team Members

1.Dhanalakshmi Naik\_\_\_\_\_\_

2.Nirupama Sundararaman

3.Khando Dechen

Date: 05/16/2016

ISTE- 724 Data Warehousing

~ Table of Contents ~

[I. Data Mart Design Definition 2](#_Toc149282576)

[1. Universe of Discourse 2](#_Toc149282577)

[2. Information Package 2](#_Toc149282578)

[II. Dimensional Model 3](#_Toc149282579)

[III. Data Staging: ETL – Data Extract File Definitions 5](#_Toc149282580)

[IV. Data Staging: ETL – Source-to-Target Mappings 8](#_Toc149282581)

[V. SQL Code – Tables & Constraints 9](#_Toc149282583)

[VI. Data Staging Activities - ETL 19](#_Toc149282584)

[1. Data Cleansing 19](#_Toc149282585)

[2. Data Transformation 21](#_Toc149282586)

[3. Table Population 25](#_Toc149282587)

[VII. End User Applications 30](#_Toc149282588)

[1. Queries 30](#_Toc149282589)

[2. Materialized Views & Summaries 31](#_Toc149282590)

VIII. Handling Slowly Changing Dimensions (SCD)………………………………….34

IX. N-M Implementation Option………………………………………………….…35

X. Reference………………………………………………………………………...37

# I. Data Mart Design Definition

## 1. Universe of Discourse

|  |
| --- |
| The data mart which monitors financial performance of a company by sales, profit by the products sold |

## 2. Information Package

Process Name: Financial Sales Analysis

Grain: Sales to a particular customer in a particular day from any three division

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DaDateDate** | **Product** | **Customer** | **Supplier** | **Junk** |
| Date | ProductID | CustomerID | SupplierID | Payment Method |
| Day of Month | Product Description | CustomerName | SupplierName | Order Method |
| Day Name of Week | Product Price1 | CustomerAddress1 | SupplierAdd1 | Shipping Method |
| Day of Week | Product Price2 | CustomerAddress2 | SupplierAdd2 |  |
| Day of Year | Product Unit Cost | CustomerTypeID | Supplier City |  |
| Week | Product TypeID | CustomerTypeName | Supplier State |  |
| Month | ProductType Description | Customer City | Supplier zip |  |
| Month Name | Product BUID | CustomerState |  |  |
| Quarter | Product BUID Name | Customer Zip |  |  |
| Year | Product BUID Abbrev |  |  |  |
| **Measured Facts:** Sales Price, Cost Price and Shipping cost, days to complete the order | | | |  |

Facts: Sales Price, Cost Price and Shipping cost, days to complete the order

3. Entity Definitions

|  |  |
| --- | --- |
| **Entity** | **Entity Definition** (*genus differentia*) |
| Customer | This entity contains all the customer who do business with three divisions TPC-W, TPC-E, PEC |
| Product | This entity consists of the product descritption, product types and the business unit they belong to along |
| Supplier | This entity consists of the details of the suppliers who supply to TPC-W,TPC-E,PEC |
| Date | This is an important entity of our data mart which has all the sales data and order date recorded for each product sales |
| Junk | This entity consists of the modes of order, modes of shipping and modes of payment used |
| Sales Fact table | This contains is a very specific well defined numeric attributes for analysis purpose |

# 

# II. Dimensional Model

Our dimensional model consists of 6 entities namely Customer, Product, Supplier, Date ,Junk and Sales fact table.

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

|  |  |  |
| --- | --- | --- |
| Business Unit | File name | Attributes and Data types |
| PEC | PECbusiness\_unit.csv | BUID (text), NAME (text), ABBREV (text) |
| PEC | PECcustomer\_type.csv | CUSTTYPEID (text), TYPENAME (text) |
| PEC | PECmanufacturingCosts.csv | Year (integer), Month (integer), prodID (integer), manufacturingCost (integer) |
| PEC | PECproduct.csv | prodid (integer), prodDescription (text), price1 (number), price2 (number), unitCost (number), supplierName (text), productTypeID (integer) |
| PEC | PECproduct\_type.csv | PRODTYPEID (integer), TYPEDESCRIPTION (text), BUID (text) |
| PEC | PECsales\_new | Invoice (integer), custID (integer), salesDate (date), prodid (integer),amt (number), qty (integer), shipMethod (text), shipCost (number), paymentMethod (text), orderMethod (text), orderDate (date), discounted (text) |
| PEC | PECcustomer.csv | custID (integer), name(text), address (text), city (text), state (text), zip (text), custtype (text) |
| TPCW | TPCWbusiness\_unit.csv | BUID (text), NAME (text), ABBREV (text) |
| TPCW | TPCWcustomer.csv | custID (integer), name (text), address (text), city (text), state (text), zip (text), custtype (text), stateabbrev (text) |
| TPCW | TPCWcustomer\_type.csv | CUSTTYPEID (text), TYPENAME (text) |
| TPCW | TPCWproduct.csv | ProductID (integer), ProductName (text), Price1 (number), Price2 (number), UnitCost (number), SupplierName (text), SupplierContact (text), SupplierAddress (text), SupplierCityANDState (text), SupplierZipCode (text), ProductTypeID (integer) |
| TPCW | TPCWproduct\_type.csv | PRODTYPEID (integer), TYPEDESCRIPTION (text), BUID (text) |
| TPCW | TPCWsales\_new.csv | Invoice (integer), custID (integer), prodID (integer), salesDate (date), amt (numbmer), qty (integer), discounted (text) |
| TPC-E | business\_unit.csv | BUID (text), NAME (text), ABBREV (text) |
| TPC-E | customer.csv | CUSTID (integer), NAME (text), ADDR1 (text), ADDR2 (text), CITY (text), STATE (text), ZIP (text), CUSTTYPEID (integer) |
| TPC-E | customer\_type.csv | CUSTTYPEID (text), TYPENAME (text) |
| TPC-E | invoice.csv | invoiceID (integer), custID (integer), salesDate |
| TPC-E | product.csv | PRODID (integer), DESCRIPTION (text), PRICE1 (number), PRICE2 (number), PRODTYPEID (integer), UNITCOST (number), SUPPLIERID (integer) |
| TPC-E | prod\_type.csv | PRODTYPEID (integer), TYPEDESCRIPTION (text) (text), BUID (text) |
| TPC-E | supplier.csv | SUPPLIERID (integer), NAME (text), ADDR1 (text), ADDR2 (text), CITY (text), STATE (text), ZIP (text) |

# 

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

Follow the same format as indicated in “The Data Warehouse ETL Toolkit” by Kimball & Caserta, Fig. 3.1 on page 60. This is available on Books 24x7. The table should be in alphabetical order table name and column name.

Logical Data Map:

1. Customer

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Target |  |  |  |  | Source |  |  |  |  |
| Table Name | Table Type | Column Name | Data type | SCD | Database | Table Name | Column Name | Data Type | Transformation |
| Customer | Dimension | customerAdd1 | Varchar(50) | 2 | FinanceDW | StagingTable | customerAdd1 | Varchar(50) | From input |
| Customer | Dimension | customerAdd2 | Varchar(50) | 2 | FinanceDW | StagingTable | customerAdd2 | Varchar(50) | From input |
| Customer | Dimension | customerCity | Varchar(10) | 2 | FinanceDW | StagingTable | customerCity | Varchar(10) | From input |
| Customer | Dimension | customerID | INT | 1 | FinanceDW | StagingTable | customerID | INT | From input |
| Customer | Dimension | customerKey | INT | 1 |  |  |  | INT | Customer look up in loaddimension.ktrname |
| Customer | Dimension | customerName | Varchar(50) | 1 | FinanceDW | StagingTable | customerName | Varchar(50) | From input |
| Customer | Dimension | customerState | Char(2) | 1 | FinanceDW | StagingTable | customerState | Char(2) | From input |
| Customer | Dimension | customerTypeId | Number | 3 | FinanceDW | StagingTable | customerTypeId | Number | Frominput |
| Customer | Dimension | customerTypeName | Varchar(50) | 3 | FinanceDW | StagingTable | customerTypeName | Varchar(50) | From input |
| Customer | Dimension | customerZip | Char(5) | 1 | FinanceDW | StagingTable | customerZip | Char(5) | From input |

1. Date

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Target |  |  |  |  | Source |  |  |  |  |
| Table Name | Table Type | Column  Name | Data type | SCD | Database | Table Name | Column Name | Data Type | Transformation |
| Date | Dimension | dateKey | INT |  |  |  | dateKey | INT | generated |
| Date | Dimension | Full date | INT |  | FinanceDW | StagingTable | orderdate | DATE | From Input file |
| Date | Dimension | Full date | DATE |  | FinanceDW | StagingTable | salesdate | DATE | From Input file |

# 

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Target |  |  |  |  | Source |  |  |  |  |
| Table Name | Table Type | Column  Name | Data type | SCD | Database | Table Name | Column Name | Data Type | Transformation |
| Junk | Dimension | junkKey | INT |  | FinanceDW | StagingTable |  | INT | Junk look up in loaddimension ktr |
| Junk | Dimension | orderMethod | VARCHAR(10) |  | FinanceDW | StagingTable | orderType | VARCHAR(10) | From Input file |
| Junk | Dimension | paymentMethod | VARCHAR(10) |  | FinanceDW | StagingTable | paymentMethod | VARCHAR(10) | From input file |
| Junk | Dimension | shippingMethod | VARCHAR(10) |  | FinanceDW | StagingTable | shippingMethod | VARCHAR(10) | from input file |

# Junk

# 

1. Product

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Target |  |  |  |  | Source |  |  |  |  |
| Table Name | Table Type | Column Name | Data type | SCD | Database | Table Name | Column Name | Data Type | Transformation |
| Product | Dimension | ProductBUIDAbbrev | VARCHAR(10) | 3 | FinanceDW | StagingTable | ProductBUIDAbbrev | VARCHAR(10) | From input |
| Product | Dimension | productBUID | INT | 3 | FinanceDW | StagingTable | productBUID | INT | From input |
| Product | Dimension | productBUIDName | VARCHAR(25) | 3 | FinanceDW | StagingTable | productBUIDName | VARCHAR(25) | From input |
| Product | Dimension | ProductDescription | VARCHAR(50) | 2 | FinanceDW | StagingTable | ProductDescription | VARCHAR(50) | From input |
| Product | Dimension | productID | INT | 1 | FinanceDW | StagingTable | productID | INT | From input |
| Product | Dimension | productKey | INT | 1 |  |  |  | INT | Product lookup in loaddimension.ktr |
| Product | Dimension | ProductPrice1 | DECIMAL(10,2) | 2 | FinanceDW | StagingTable | ProductPrice1 | DECIMAL(10,2) | From input |
| Product | Dimension | ProductPrice2 | DECIMAL(10,2) | 2 | FinanceDW | StagingTable | ProductPrice2 | DECIMAL(10,2) | From input |
| Product | Dimension | productTypeDescription | VARCHAR(25) | 3 | FinanceDW | StagingTable | productTypeDescription | VARCHAR(25) | From input |
| Product | Dimension | productTypeID | INT | 3 | FinanceDW | StagingTable | productTypeID | INT | From input |
| Product | Dimension | productUnitCost | DECIMAL(10,2) | 2 | FinanceDW | StagingTable | productUnitCost | DECIMAL(10,2) |  |

1. SalesFact

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Target |  |  |  |  | Source |  |  |  |  |
| Table Name | Table Type | Column  Name | Data type | SCD | Database | Table Name | Column Name | Data Type | Transformation |
| SALESFACT | FACT | costPrice | Dec(10,2) |  | FinanceDW | StagingTable | costPrice | Dec(10,2) | Cost Price = amt \* qty |
| SALESFACT | FACT | customerKey | INT |  |  | Customer |  | INT | Ex: Customer Lookup in FACTLOAD ktr |
| SALES FACT | FACT | daysToShip | Number |  | FinaceDW | StagingTable | daysToShip | Number | saleDate-orderDate |
| SALESFACT | FACT | division | Varchar |  | FinanceDW | StagingTable | division |  |  |
| SALESFACT | FACT | junkKey | INT |  |  | Junk |  | INT | Junk Lookup in FACTLOAD ktr |
| SALESFACT | FACT | orderdateKey | INT |  |  | Date |  | INT | orderdate Lookup in FACTLOAD .ktr |
| SALESFACT | FACT | productKey | INT |  |  | Product |  | INT | Product Lookup in FACTLOAD.ktr |
| SALESFACT | FACT | quantity | INT |  | FinanceDW | StagingTable | quantity | INT |  |
| SALESFACT | FACT | saledateKey | INT |  |  | Date |  | INT | salesdate Lookup in FACTLOAD ktr |
| SALESFACT | FACT | salesPrice | Dec(10,2) |  | FinanceDW | StagingTable | salesCost | Dec(10,2) |  |
| SALES FACT | FACT | shippingCost | decimal |  | FINANceDW | StagingTable | shippingCost |  |  |
| SALESFACT | FACT | supplierKey | INT |  |  | Supplier |  | INT | supplier Lookup in FACTLOAD ktr |

1. Supplier

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Target |  |  |  |  | Source |  |  |  |  |
| Table Name | Table Type | Column  Name | Data type | SCD | Database | Table Name | Column Name | Data Type | Transformation |
| Supplier | Dimension | supplierAdd1 | VARCHAR(50) | 2 | FinanceDW | StagingTable | supplierAdd1 | VARCHAR(50) | From input |
| Supplier | Dimension | supplierAdd2 | VARCHAR(50) | 2 | FinanceDW | StagingTable | supplierAdd2 | VARCHAR(50 | From input |
| Supplier | Dimension | supplierCity | VARCHAR(20) | 2 | FinanceDW | StagingTable | supplierCity | VARCHAR(20) | From input |
| Supplier | Dimension | supplierID | INT | 1 | FinanceDW | StagingTable | supplierID | INT | From input |
| Supplier | Dimension | supplierKey | INT | 1 |  |  |  | INT | Supplier lookup in Loaddimensionktr |
| Supplier | Dimension | supplierName | VARCHAR(50) |  | FinanceDW | StagingTable | supplierName | VARCHAR(50) | From input |
| Supplier | Dimension | supplierState | CHAR(2) | 2 | FinanceDW | StagingTable | supplierState | CHAR(2) | From input |

# V. SQL Code – Tables & Constraints

CREATE SCHEMA IF NOT EXISTS `FINANCEDW` DEFAULT CHARACTER SET utf8 ;

USE `FINANCEDW` ;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`Customer`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`Customer` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`Customer` (

`customerKey` INT(11) NOT NULL AUTO\_INCREMENT COMMENT '',

`customerID` INT(11) NULL COMMENT '',

`customerName` VARCHAR(50) NULL COMMENT '',

`customerAdd1` VARCHAR(50) NULL COMMENT '',

`customerAdd2` VARCHAR(50) NULL COMMENT '',

`customerCity` VARCHAR(20) NULL COMMENT '',

`customerState` CHAR(2) NULL COMMENT '',

`customerZip` CHAR(5) NULL COMMENT '',

`customerTypeID` CHAR(1) NULL COMMENT '',

`customerTypeName` VARCHAR(20) NULL COMMENT '',

PRIMARY KEY (`customerKey`) COMMENT '',

UNIQUE INDEX `customerKey\_UNIQUE` (`customerKey` ASC) COMMENT '')

ENGINE = InnoDB

AUTO\_INCREMENT = 64

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`Date`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`Date` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`Date` (

`dateKey` INT(11) NOT NULL AUTO\_INCREMENT COMMENT '',

`fullDate` DATETIME NULL DEFAULT NULL COMMENT '',

`dateName` CHAR(11) NULL DEFAULT NULL COMMENT '',

`dateNameUS` CHAR(11) NULL DEFAULT NULL COMMENT '',

`dateNameEU` CHAR(11) NULL DEFAULT NULL COMMENT '',

`dayOfweek` TINYINT(10) NULL DEFAULT NULL COMMENT '',

`dayNameOfWeek` CHAR(10) NULL DEFAULT NULL COMMENT '',

`dayOfMonth` TINYINT(10) NULL DEFAULT NULL COMMENT '',

`dayOfYear` SMALLINT(10) NULL DEFAULT NULL COMMENT '',

`weekdayWeekend` CHAR(7) NULL DEFAULT NULL COMMENT '',

`weekOfYear` TINYINT(10) NULL DEFAULT NULL COMMENT '',

`monthName` CHAR(10) NULL DEFAULT NULL COMMENT '',

`monthOfYear` TINYINT(10) NULL DEFAULT NULL COMMENT '',

`isLastDayOfMonth` CHAR(1) NULL DEFAULT NULL COMMENT '',

`calendarQuarter` TINYINT(10) NULL DEFAULT NULL COMMENT '',

`calendarYear` SMALLINT(10) NULL DEFAULT NULL COMMENT '',

`calendarYearMonth` CHAR(7) NULL DEFAULT NULL COMMENT '',

`calendarYearQtr` CHAR(7) NULL DEFAULT NULL COMMENT '',

`fiscalMonthOfYear` TINYINT(10) NULL DEFAULT NULL COMMENT '',

`fiscalQuarter` TINYINT(10) NULL DEFAULT NULL COMMENT '',

`fiscalYear` INT(11) NULL DEFAULT NULL COMMENT '',

`fiscalYearMonth` CHAR(9) NULL DEFAULT NULL COMMENT '',

`fiscalYearQtr` CHAR(8) NULL DEFAULT NULL COMMENT '',

PRIMARY KEY (`dateKey`) COMMENT '')

ENGINE = InnoDB

AUTO\_INCREMENT = 20201232

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`Junk`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`Junk` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`Junk` (

`junkKey` INT(11) NOT NULL AUTO\_INCREMENT COMMENT '',

`shippingMethod` VARCHAR(10) NULL DEFAULT NULL COMMENT '',

`orderMethod` VARCHAR(10) NULL DEFAULT NULL COMMENT '',

`paymentMethod` VARCHAR(10) NULL DEFAULT NULL COMMENT '',

PRIMARY KEY (`junkKey`) COMMENT '',

UNIQUE INDEX `junkKey\_UNIQUE` (`junkKey` ASC) COMMENT '')

ENGINE = InnoDB

AUTO\_INCREMENT = 29

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`Product`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`Product` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`Product` (

`productKey` INT(11) NOT NULL AUTO\_INCREMENT COMMENT '',

`productID` INT(11) NULL DEFAULT NULL COMMENT '',

`productDescription` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`productPrice1` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`productPrice2` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`productUnitCost` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`productTypeID` INT(11) NULL DEFAULT NULL COMMENT '',

`productTypeDescription` VARCHAR(25) NULL DEFAULT NULL COMMENT '',

`productBUID` CHAR(1) NULL DEFAULT NULL COMMENT '',

`productBUIDName` VARCHAR(25) NULL DEFAULT NULL COMMENT '',

`productBUIDAbbrev` VARCHAR(10) NULL DEFAULT NULL COMMENT '',

PRIMARY KEY (`productKey`) COMMENT '',

UNIQUE INDEX `productKey\_UNIQUE` (`productKey` ASC) COMMENT '')

ENGINE = InnoDB

AUTO\_INCREMENT = 64

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`Supplier`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`Supplier` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`Supplier` (

`supplierKey` INT(11) NOT NULL AUTO\_INCREMENT COMMENT '',

`supplierID` INT(11) NULL DEFAULT NULL COMMENT '',

`supplierName` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierAdd1` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierAdd2` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierCity` VARCHAR(20) NULL DEFAULT NULL COMMENT '',

`supplierState` CHAR(2) NULL DEFAULT NULL COMMENT '',

`supplierZip` CHAR(5) NULL DEFAULT NULL COMMENT '',

PRIMARY KEY (`supplierKey`) COMMENT '',

UNIQUE INDEX `supplierKey\_UNIQUE` (`supplierKey` ASC) COMMENT '')

ENGINE = InnoDB

AUTO\_INCREMENT = 16

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`SalesFact`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`SalesFact` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`SalesFact` (

`customerKey` INT(11) NOT NULL COMMENT '',

`productKey` INT(11) NOT NULL COMMENT '',

`supplierKey` INT(11) NOT NULL COMMENT '',

`orderdateKey` INT(11) NOT NULL COMMENT '',

`junkKey` INT(11) NOT NULL COMMENT '',

`saledateKey` INT(11) NOT NULL COMMENT '',

`salesPrice` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`costPrice` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`quantity` INT(11) NULL DEFAULT NULL COMMENT '',

`division` VARCHAR(45) NOT NULL COMMENT '',

`invoiceID` INT(11) NOT NULL COMMENT '',

`daysToShip` VARCHAR(45) NULL COMMENT '',

`shippingcost` VARCHAR(45) NULL COMMENT '',

PRIMARY KEY (`customerKey`, `productKey`, `supplierKey`, `orderdateKey`, `junkKey`, `saledateKey`, `division`, `invoiceID`) COMMENT '')

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`sales staging table`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`sales staging table` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`sales staging table` (

`customerID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`customerName` VARCHAR(50) NULL COMMENT '',

`custAdd1` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`custAddr2` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`custState` CHAR(2) NULL DEFAULT NULL COMMENT '',

`custCity` VARCHAR(20) NULL COMMENT '',

`custZip` CHAR(5) NULL DEFAULT NULL COMMENT '',

`custTypeID` CHAR(1) NULL DEFAULT NULL COMMENT '',

`custTypeName` VARCHAR(20) NULL DEFAULT NULL COMMENT '',

`shippingMethod` VARCHAR(10) NOT NULL DEFAULT NULL COMMENT '',

`orderType` VARCHAR(10) NOT NULL DEFAULT NULL COMMENT '',

`paymentMethod` VARCHAR(10) NOT NULL DEFAULT NULL COMMENT '',

`prodID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`prodDescription` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`price1` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`price2` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`unitCost` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`prodTypeDescription` VARCHAR(25) NULL DEFAULT NULL COMMENT '',

`prodTypeID` INT(11) NULL DEFAULT NULL COMMENT '',

`prodBUID` CHAR(1) NULL DEFAULT NULL COMMENT '',

`BUIDName` VARCHAR(25) NULL DEFAULT NULL COMMENT '',

`abbrev` VARCHAR(10) NULL DEFAULT NULL COMMENT '',

`supplierID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`supplierName` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierAddr1` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierCity` VARCHAR(20) NULL DEFAULT NULL COMMENT '',

`supplierAddr2` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierState` CHAR(2) NULL DEFAULT NULL COMMENT '',

`supplierZip` CHAR(5) NULL DEFAULT NULL COMMENT '',

`salesDate` DATE NOT NULL DEFAULT NULL COMMENT '',

`orderDate` DATE NOT NULL DEFAULT NULL COMMENT '',

`division` VARCHAR(45) NOT NULL DEFAULT NULL COMMENT '',

`quantity` INT(11) NULL DEFAULT NULL COMMENT '',

`saleCost` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`invoiceID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`costPrice` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`daytoShip` VARCHAR(45) NULL COMMENT '',

`shippingcost` VARCHAR(45) NULL COMMENT '',

`sales staging tablecol` VARCHAR(45) NULL COMMENT '',

PRIMARY KEY (`customerID`, `shippingMethod`, `orderType`, `paymentMethod`, `prodID`, `salesDate`, `invoiceID`, `division`, `supplierID`, `orderDate`) COMMENT '')

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `FINANCEDW`.`StagingTable`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `FINANCEDW`.`StagingTable` ;

CREATE TABLE IF NOT EXISTS `FINANCEDW`.`StagingTable` (

`customerID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`customerName` VARCHAR(50) NULL COMMENT '',

`customerAdd1` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`customerAdd2` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`customerState` CHAR(2) NULL DEFAULT NULL COMMENT '',

`customerCity` VARCHAR(20) NULL COMMENT '',

`customerZip` CHAR(5) NULL DEFAULT NULL COMMENT '',

`customerTypeID` CHAR(1) NULL DEFAULT NULL COMMENT '',

`customerTypeName` VARCHAR(20) NULL DEFAULT NULL COMMENT '',

`shippingMethod` VARCHAR(10) NOT NULL DEFAULT NULL COMMENT '',

`orderMethod` VARCHAR(10) NOT NULL DEFAULT NULL COMMENT '',

`paymentMethod` VARCHAR(10) NOT NULL DEFAULT NULL COMMENT '',

`productID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`productDescription` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`productPrice1` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`productPrice2` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`productUnitCost` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`productTypeDescription` VARCHAR(25) NULL DEFAULT NULL COMMENT '',

`productTypeID` INT(11) NULL DEFAULT NULL COMMENT '',

`productBUID` CHAR(1) NULL DEFAULT NULL COMMENT '',

`productBUIDName` VARCHAR(25) NULL DEFAULT NULL COMMENT '',

`productBUIDAbbrev` VARCHAR(10) NULL DEFAULT NULL COMMENT '',

`supplierID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`supplierName` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierAdd1` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierCity` VARCHAR(20) NULL DEFAULT NULL COMMENT '',

`supplierAdd2` VARCHAR(50) NULL DEFAULT NULL COMMENT '',

`supplierState` CHAR(2) NULL DEFAULT NULL COMMENT '',

`supplierZip` CHAR(5) NULL DEFAULT NULL COMMENT '',

`saleDate` DATE NOT NULL DEFAULT NULL COMMENT '',

`orderDate` DATE NOT NULL DEFAULT NULL COMMENT '',

`division` VARCHAR(45) NOT NULL DEFAULT NULL COMMENT '',

`quantity` INT(11) NULL DEFAULT NULL COMMENT '',

`saleCost` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`invoiceID` INT(11) NOT NULL DEFAULT NULL COMMENT '',

`costPrice` DECIMAL(10,2) NULL DEFAULT NULL COMMENT '',

`dayToShip` VARCHAR(45) NULL COMMENT '',

PRIMARY KEY (`customerID`, `shippingMethod`, `orderMethod`, `paymentMethod`, `productID`, `saleDate`, `invoiceID`, `division`, `supplierID`, `orderDate`) COMMENT '')

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

USE `FINANCEDW` ;

-- -----------------------------------------------------

-- procedure PopulateDateDimension

-- -----------------------------------------------------

USE `FINANCEDW`;

DROP procedure IF EXISTS `FINANCEDW`.`PopulateDateDimension`;

DELIMITER $$

USE `FINANCEDW`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `PopulateDateDimension`(BeginDate DATETIME, EndDate DATETIME)

BEGIN

# =============================================

# Description: http://arcanecode.com/2009/11/18/populating-a-kimball-date-dimension/

# =============================================

# A few notes, this code does nothing to the existing table, no deletes

# are triggered before hand. Because the DateKey is uniquely indexed,

# it will simply produce errors if you attempt to insert duplicates.

# You can however adjust the Begin/End dates and rerun to safely add

# new dates to the table every year.

#

# If the begin date is after the end date, no errors occur but nothing

# happens as the while loop never executes.

# Holds a flag so we can determine if the date is the last day of month

DECLARE LastDayOfMon CHAR(1);

# Number of months to add to the date to get the current Fiscal date

DECLARE FiscalYearMonthsOffset INT;

# These two counters are used in our loop.

DECLARE DateCounter DATETIME; #Current date in loop

DECLARE FiscalCounter DATETIME; #Fiscal Year Date in loop

# Set this to the number of months to add to the current date to get

# the beginning of the Fiscal year. For example, if the Fiscal year

# begins July 1, put a 6 there.

# Negative values are also allowed, thus if your 2010 Fiscal year

# begins in July of 2009, put a -6.

SET FiscalYearMonthsOffset = 6;

# Start the counter at the begin date

SET DateCounter = BeginDate;

WHILE DateCounter <= EndDate DO

# Calculate the current Fiscal date as an offset of

# the current date in the loop

SET FiscalCounter = DATE\_ADD(DateCounter, INTERVAL FiscalYearMonthsOffset MONTH);

# Set value for IsLastDayOfMonth

IF MONTH(DateCounter) = MONTH(DATE\_ADD(DateCounter, INTERVAL 1 DAY)) THEN

SET LastDayOfMon = 'N';

ELSE

SET LastDayOfMon = 'Y';

END IF;

# add a record into the date dimension table for this date

INSERT INTO Date

(dateKey

,fullDate

,dateName

,dateNameUS

,dateNameEU

,dayOfweek

,dayNameOfWeek

,dayOfMonth

,dayOfYear

,weekdayWeekend

,weekOfYear

,monthName

,monthOfYear

,isLastDayOfMonth

,calendarQuarter

,calendarYear

,calendarYearMonth

,calendarYearQtr

,fiscalMonthOfYear

,fiscalQuarter

,fiscalYear

,fiscalYearMonth

,fiscalYearQtr)

VALUES (

( YEAR(DateCounter) \* 10000 ) + ( MONTH(DateCounter)

\* 100 )

+ DAY(DateCounter) #DateKey

, DateCounter # FullDate

, CONCAT(CAST(YEAR(DateCounter) AS CHAR(4)),'/',DATE\_FORMAT(DateCounter,'%m'),'/',DATE\_FORMAT(DateCounter,'%d')) #DateName

, CONCAT(DATE\_FORMAT(DateCounter,'%m'),'/',DATE\_FORMAT(DateCounter,'%d'),'/',CAST(YEAR(DateCounter) AS CHAR(4)))#DateNameUS

, CONCAT(DATE\_FORMAT(DateCounter,'%d'),'/',DATE\_FORMAT(DateCounter,'%m'),'/',CAST(YEAR(DateCounter) AS CHAR(4)))#DateNameEU

, DAYOFWEEK(DateCounter) #DayOfWeek

, DAYNAME(DateCounter) #DayNameOfWeek

, DAYOFMONTH(DateCounter) #DayOfMonth

, DAYOFYEAR(DateCounter) #DayOfYear

, CASE DAYNAME(DateCounter)

WHEN 'Saturday' THEN 'Weekend'

WHEN 'Sunday' THEN 'Weekend'

ELSE 'Weekday'

END #WeekdayWeekend

, WEEKOFYEAR(DateCounter) #WeekOfYear

, MONTHNAME(DateCounter) #MonthName

, MONTH(DateCounter) #MonthOfYear

, LastDayOfMon #IsLastDayOfMonth

, QUARTER(DateCounter) #CalendarQuarter

, YEAR(DateCounter) #CalendarYear

, CONCAT(CAST(YEAR(DateCounter) AS CHAR(4)),'-',DATE\_FORMAT(DateCounter,'%m')) #CalendarYearMonth

, CONCAT(CAST(YEAR(DateCounter) AS CHAR(4)),'Q',QUARTER(DateCounter)) #CalendarYearQtr

, MONTH(FiscalCounter) #[FiscalMonthOfYear]

, QUARTER(FiscalCounter) #[FiscalQuarter]

, YEAR(FiscalCounter) #[FiscalYear]

, CONCAT(CAST(YEAR(FiscalCounter) AS CHAR(4)),'-',DATE\_FORMAT(FiscalCounter,'%m')) #[FiscalYearMonth]

, CONCAT(CAST(YEAR(FiscalCounter) AS CHAR(4)),'Q',QUARTER(FiscalCounter)) #[FiscalYearQtr]

);

# Increment the date counter for next pass thru the loop

SET DateCounter = DATE\_ADD(DateCounter, INTERVAL 1 DAY);

END WHILE;

END$$

DELIMITER ;

# 

# VI. Data Staging Activities - ETL

## 1. Data Cleansing

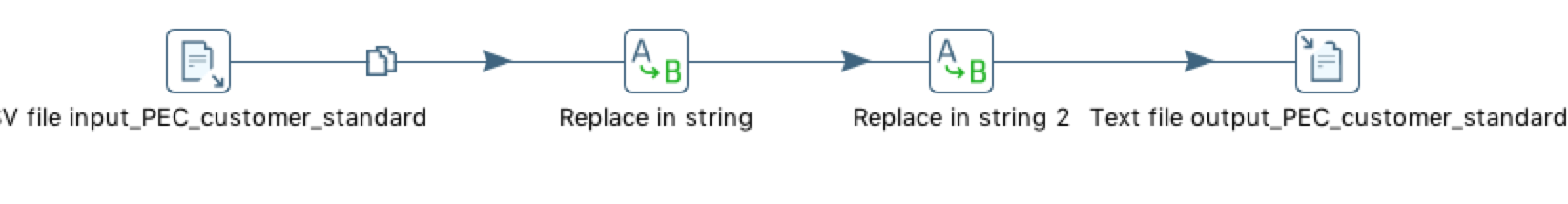
|  |  |  |  |
| --- | --- | --- | --- |
| **DM Table** | **Attribute** | **Problem** | **Resolution Strategy** |
| PECcustomer.csv | address | Need standardization | We split the address into addr1 and addr2. Changed Rd=Road, Ave.=Avenue, St= Street(PEC\_CUSTOMER\_STANDARDIZE.ktr) Fig 1 |
| PECcustomer.csv | city | Need standardization | Standardized with their abbreviations(PEC\_CUSTOMER\_STANDARDIZE.ktr) Fig 1 |
|  | state | Need standardization | Standardized with their abbreviations(PEC\_CUSTOMER\_STANDARDIZE.ktr) fig 1 |
| PECcustomer.csv | custtype | Need to Standardize | US GOV to US GOVT (PEC\_CUSTOMER\_STANDARDIZE.ktr)Fig 1 |
| PECsales\_new.csv | orderdate | Dates are in different format | Standardized date to mm/dd/yyyy format in excel |
| PECsales\_new.csv | salesdate | Dates are in different format | Standardized date to mm/dd/yyyy format in excel |
| PECproduct.csv | suppliename | null | Set it to PEC(PEC\_PRODUCT\_SUPPLIERNAME.ktr) Fig 2 |
| PECproduct.csv | unitcost | null | If the discounted value was 0 then unit cost = price1\*qty else the unit cost= price2\*qty, calculated it using pentaho(UnitCost.ktr) fig 3 |
| PECsales\_new.csv | Shipping Method | Truck was spelled as tuck | Changed it to truck using excel |
| PECManufacturingCost.csv | Year | Incorrect format | Changed to YYYY format using excel |
| PECManufacturingCost.csv | Month | Incorrect format | Changed to MM format using excel |
| PECsales\_new.csv | custID | Swap fields of row 41002 , 20000 | Shifted columns s |
| PECbusiness\_unit.csv | BUID | Double quotes enclosure | Replaced double quote with space in the field using pentaho(PEC\_Business\_Unit.ktr)Fig 4 |
| PECbusiness\_unit.csv | NAME | Double quotes enclosure | Replaced double quote with space in the field pentaho(PEC\_Business\_Unit.ktr)Fig 4 |
| PECbusiness\_unit.csv | ABBREV | Double quotes enclosure | Replaced double quote with space in the field using pentaho(PEC\_Business\_Unit.ktr) Fig 4 |
| PECbusiness\_unit.csv | abbrev | Missing value(BUID=D) | missing value was left blank |
| PECbusiness\_unit.csv | BUID | Extra space | Trim the right space |
| TPCWcustomer.csv | Address | Need standardization | St=street, Ln=Lane, Ave=Avenue, Rd=Road, Dr= Drive using replace string in pentaho(TPCW\_CustomerStandradize.ktr) |
| TPCWcustomer.csv | city | Need standardization | Spellings were corrected(TPCW\_CustomerStandradize.ktr) |
| TPCWcustomer.csv | state | Need standardization | Standardized with their abbreviations(TPCW\_CustomerStandradize.ktr) |
|  | custype | Need Standardization | We replaced the custype abbrev with custypename(TPCW\_CustomerStandradize.ktr) |
| TPCWbusiness\_unit.csv | Abbrev | Missing value(BUID=D) | Kept it as it is |
| TPCWbusiness\_unit.csv | abbrev | Missing value(BUID=D) | Kept it as it is |
|  |  |  |  |
| TPCWsales\_new.csv | custID | custID was -1,  The entire header row was getting repeated. (row number= 42815) | Set CustID=9999  Removed the entire row in which the header was repeated as it had no data. |
| TPCWsales\_new.csv | invoideID | Row with only invoiceID present | Deleted it in excel |
| TPCWsales\_new.csv | salesdate | Dates are in different format | Changed to mm/dd/yyyy format |
| TPCWproduct\_type.csv | Product type ID | The ID’s had 000 before the number, for example: 9 was written as 0009 | Removed the zeros using pentaho. (TPCW\_ProductType\_Std.ktr) fig 5 |
| TPCWproduct.csv | 1. Supplier city  2. Address  3. Supplier city | Supplier city and state were in the same column  Supplier address2 needed standardization  Spelling of the city was wrong | Divided the supplier city and supplier sate to two different columns  Standardized it.  Fixed it using pentaho. |
| TPCWproduct.csv | productID | Duplication of products due to different address 2 values | Standardized it and fixed it using pentaho fig 7 |

## 2. Data Transformation

Customer Dimension table transformation:

For customer data both from TPC and PEC we denormalized Customer Type table and Customer table and checked whether if the rows from various sources are overlapped with each other if so we eliminated any duplicate data and made conformed customer dimension using pentaho and excel. We mapped the pentaho table data types to MySQL datatype. After transforming we loaded the data into MYSQL database.

Fig 1:



Product Table Transformation**:**

After cleansing the data, we computed salescost from the given Price 1 and Price2 from the product data. We also calculated the manufacturing cost using Pentaho. The following screen shot of Pentaho depicts the product transformations:

Fig 2:

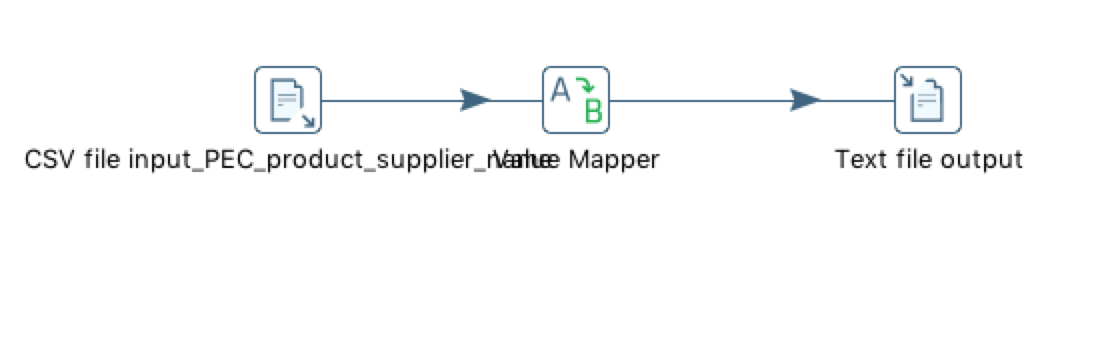


Fig 3:

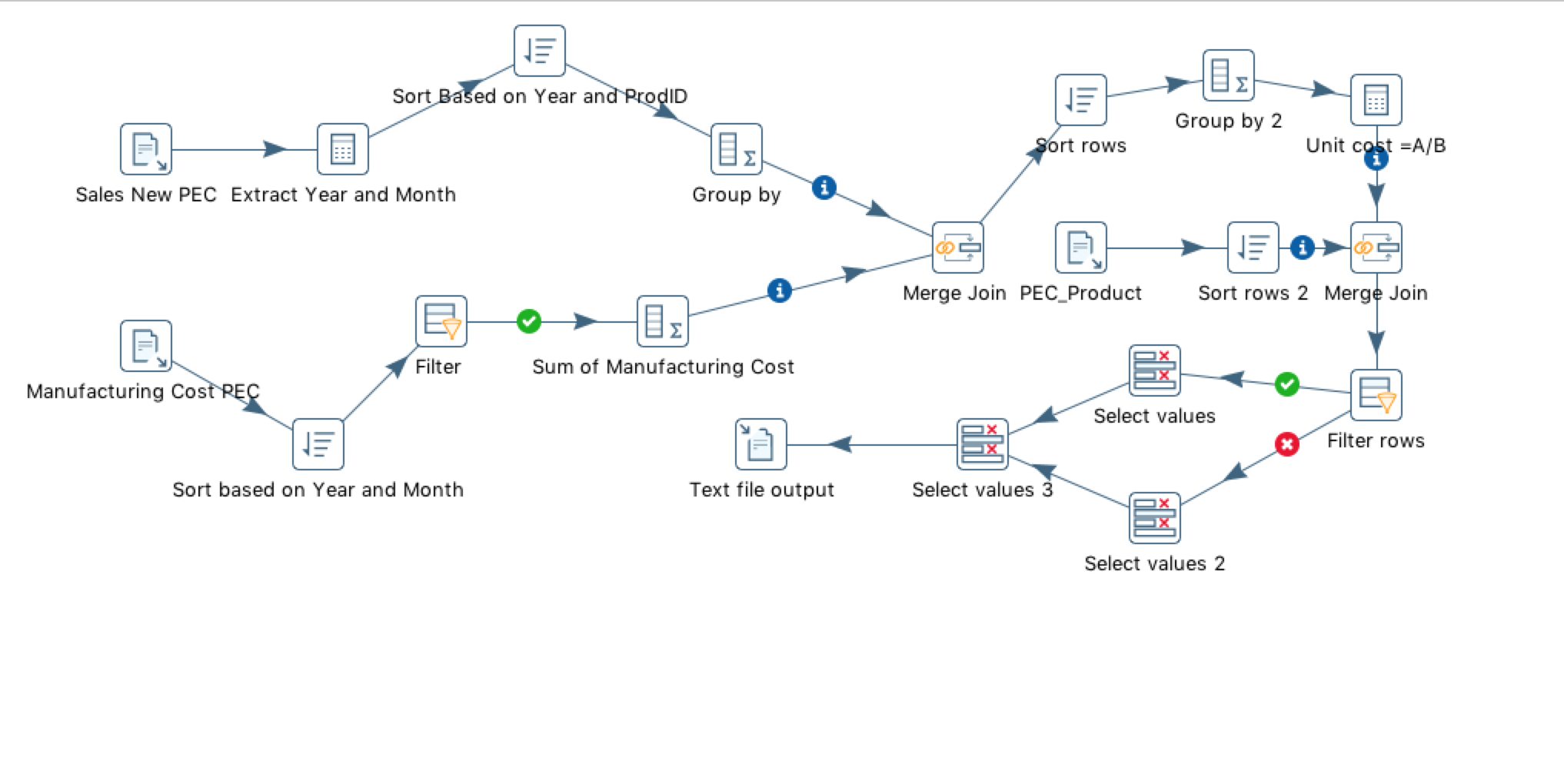


Fig 4:

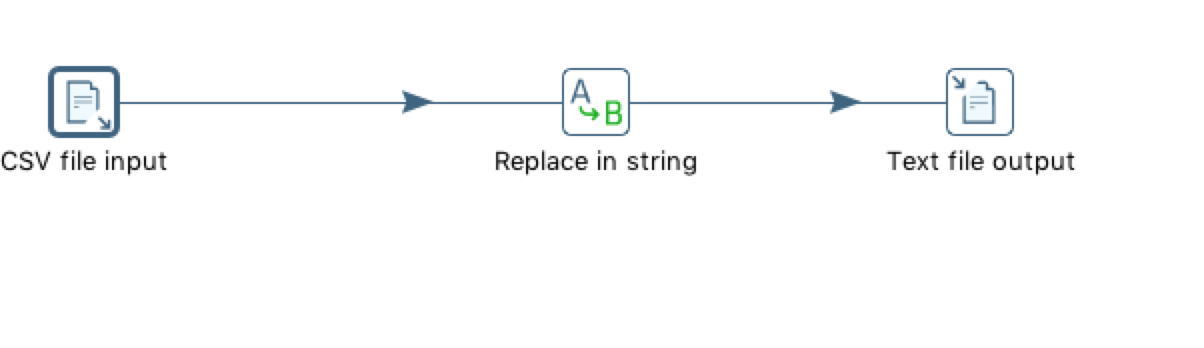


Fig 5:

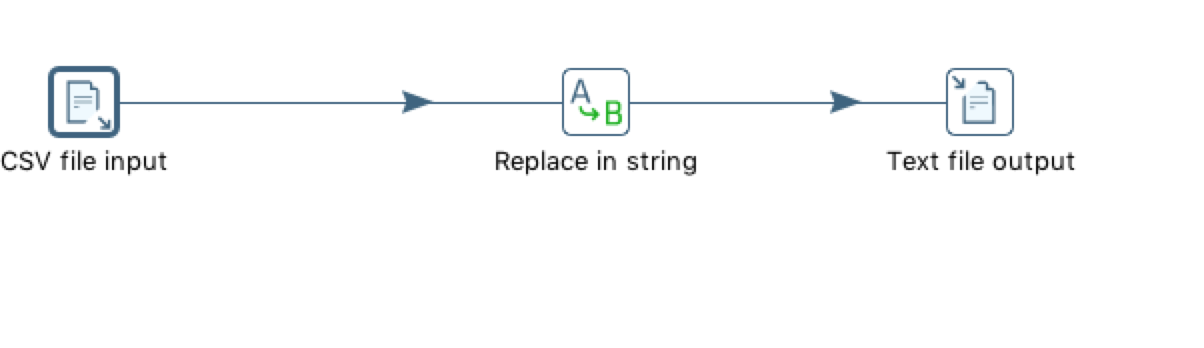


Fig 6:

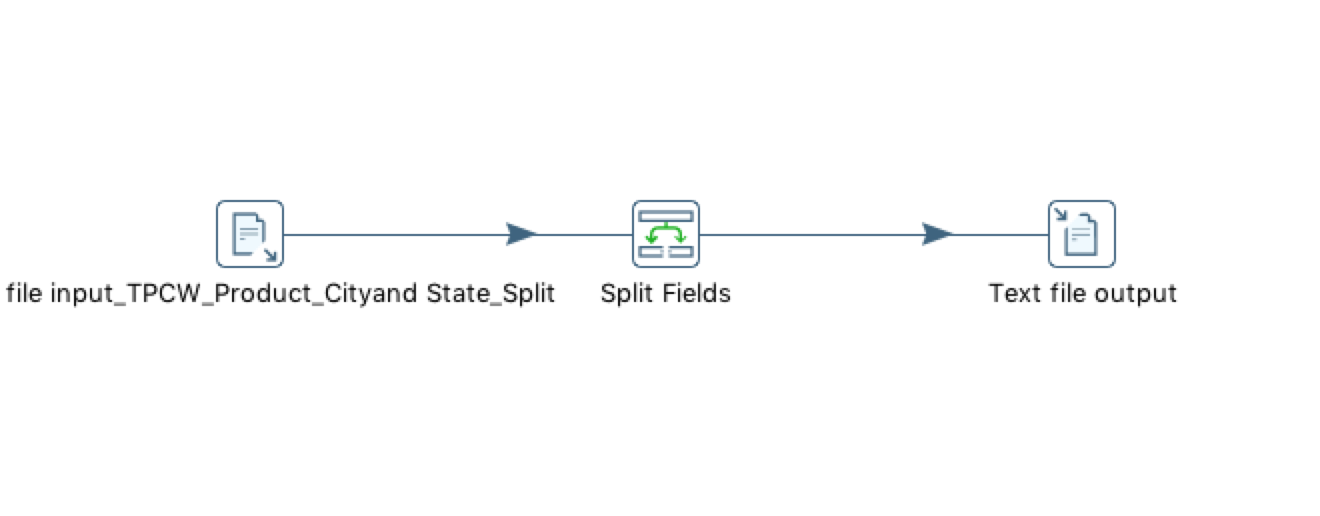
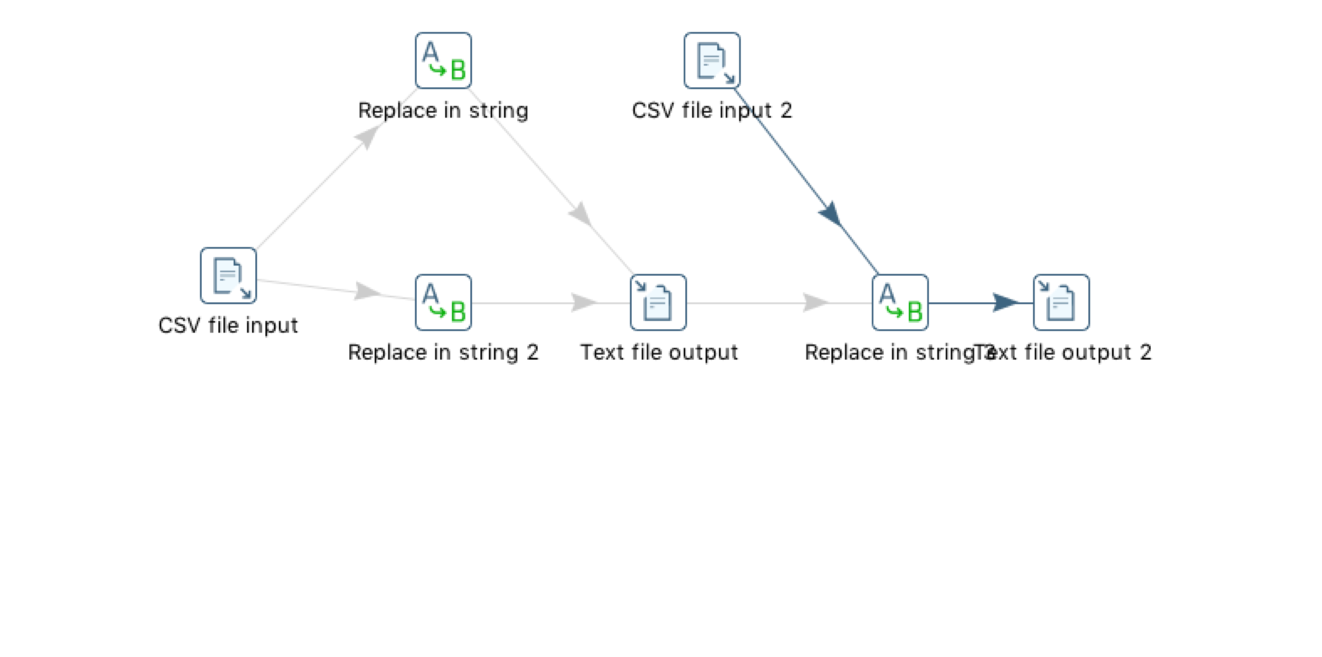
****

Fig 7:

****

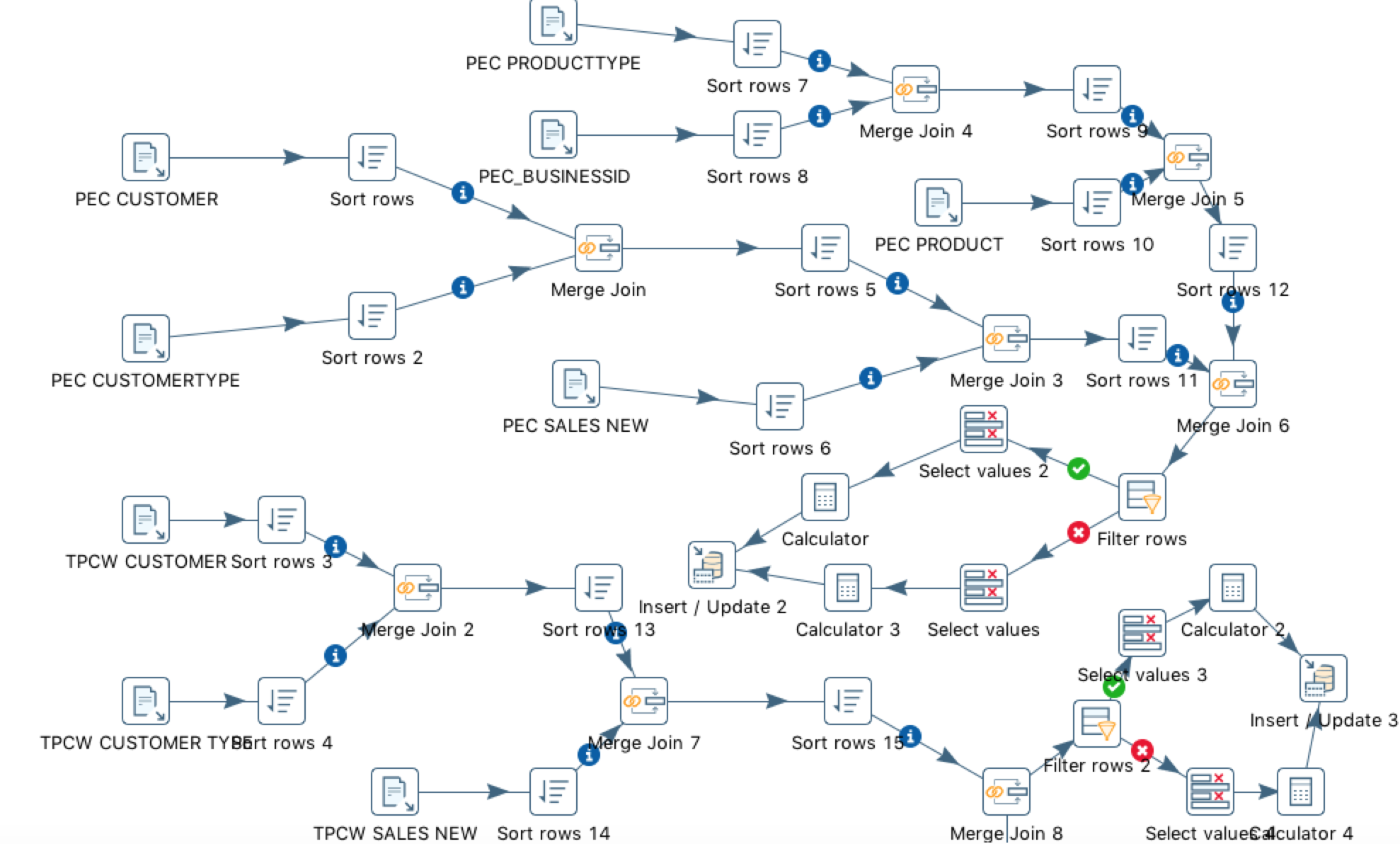
Supplier Table Transformation:

Transformation for this table was included along with the Product transformation and loaded to the staging table.

Junk table Transformation:

No transformation was made for this table. Data from PECsales\_new.csv and TPCWsales.new.csv were loaded directly into the staging table and on to the dimension table.

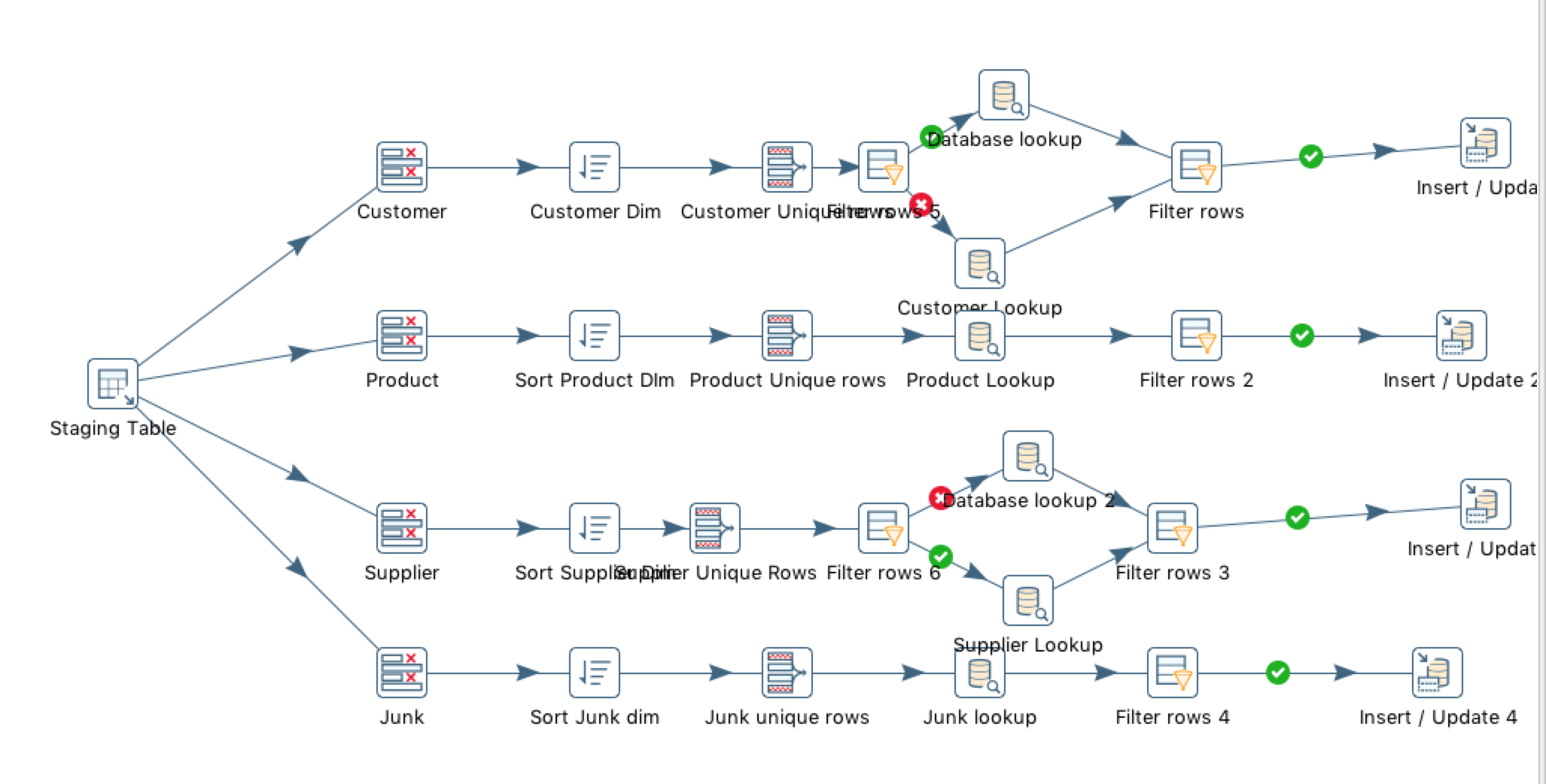
All the transformed data was loaded to the staging table called sales staging table from which was used for further population of the tables. The following transformation was used to load the staging table:



## 3. Table Population

|  |  |
| --- | --- |
| **DM Table** | **Table population process** |
| Customer Dimension | Customer Dim consist of surrogate key which refers to Fact table |
| Sales Fact Table | Fact Table consist of facts and composite primary key which are combination of all surrogate key |
| Product Dimension | Product Dim consist of surrogate key which refers to Fact table |
| Junk Dimension | Junk Dim consist of surrogate key which refers to Fact table |
| Date Dimension | Date Dim consist of surrogate key which refers to Fact table |
| Supplier Dimension | Supplier dimension consists of surrogate key which refers to Fact table |

Following Pentaho transformation was used to populates out dimensions tables from the staging table:



After cleaning the data, we imported data to the Sales fact using our Staging Table. The cost price of the staging table is the manufacturing cost. The loading of the data was done using pentaho and the transformations are as follows:

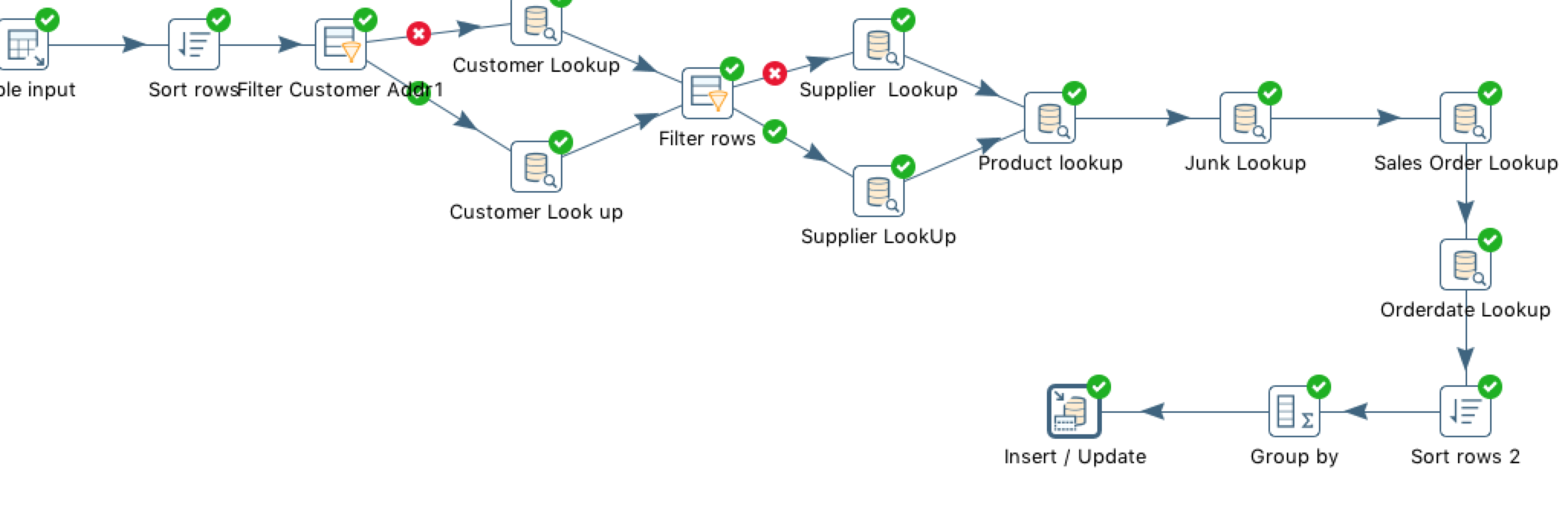
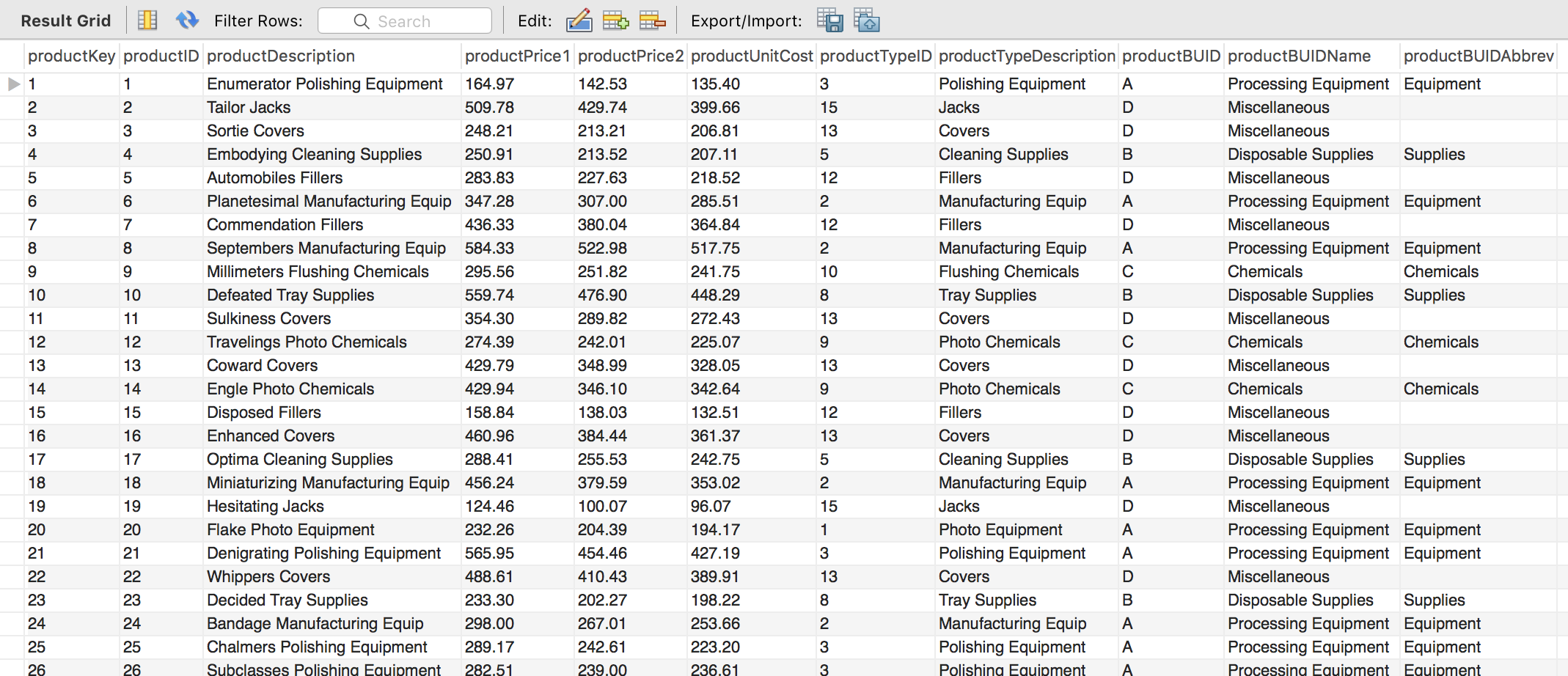


Table views after loading the dimensions are given below:

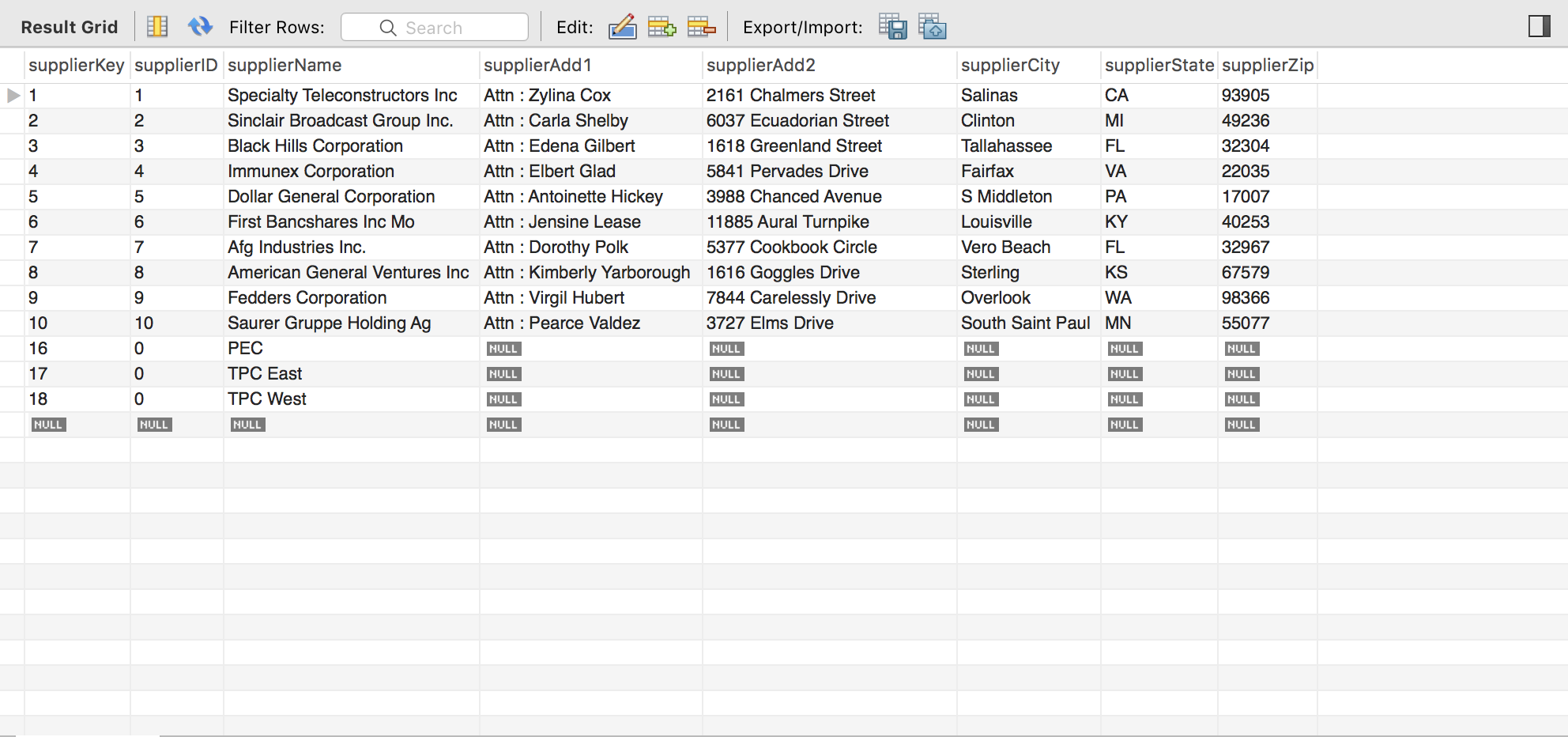
Customer



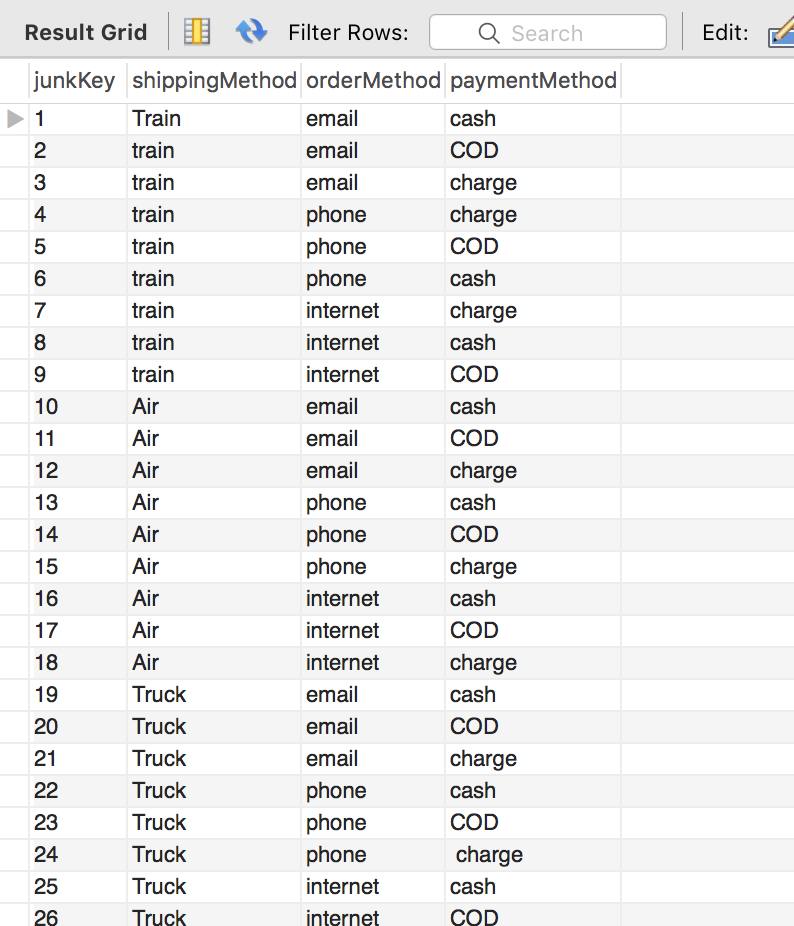
Product:



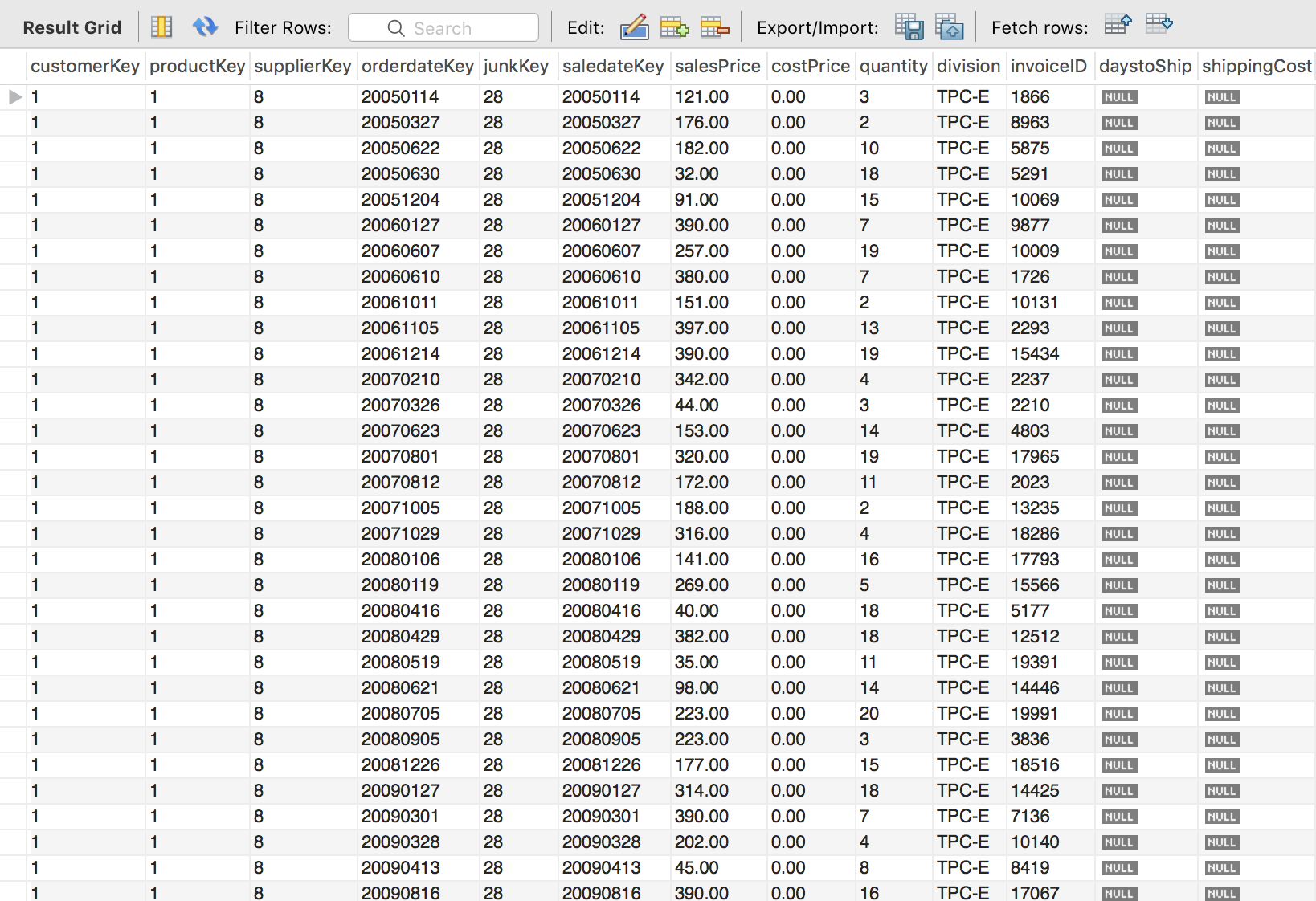
Supplier:



Junk:



SalesFact:



# VII. End User Applications

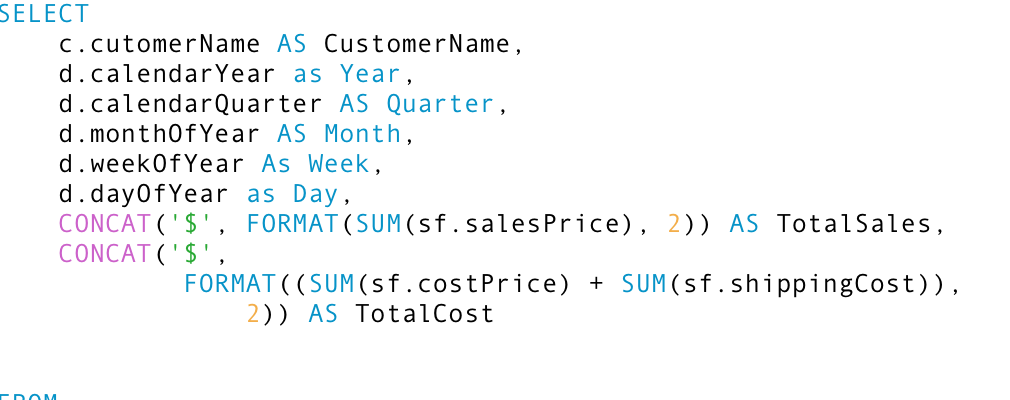
## 1. Queries

|  |
| --- |
| **User Question/Reporting Need** |
| A report that shows the sales, and costs associated with each customer or customer type on an annual, quarterly, monthly or weekly basis. |
| **SQL Code** |
| use FINANCEDW;  SELECT  c.cutomerName AS CustomerName,  d.calendarYear as Year,  d.calendarQuarter AS Quarter,  d.monthOfYear AS Month,  d.weekOfYear As Week,  d.dayOfYear as Day,  CONCAT('$', FORMAT(SUM(sf.salesPrice), 2)) AS TotalSales,  CONCAT('$',  FORMAT((SUM(sf.costPrice) + SUM(sf.shippingCost)),  2)) AS TotalCost      FROM  SalesFact sf  JOIN  Customer c ON sf.customerKey = c.customerKey  JOIN  Date d ON sf.saledateKey = d.dateKey  GROUP BY CustomerName , Year , Quarter, Month, Week, Day  ; |
| **Supporting Index(es)** |
|  |

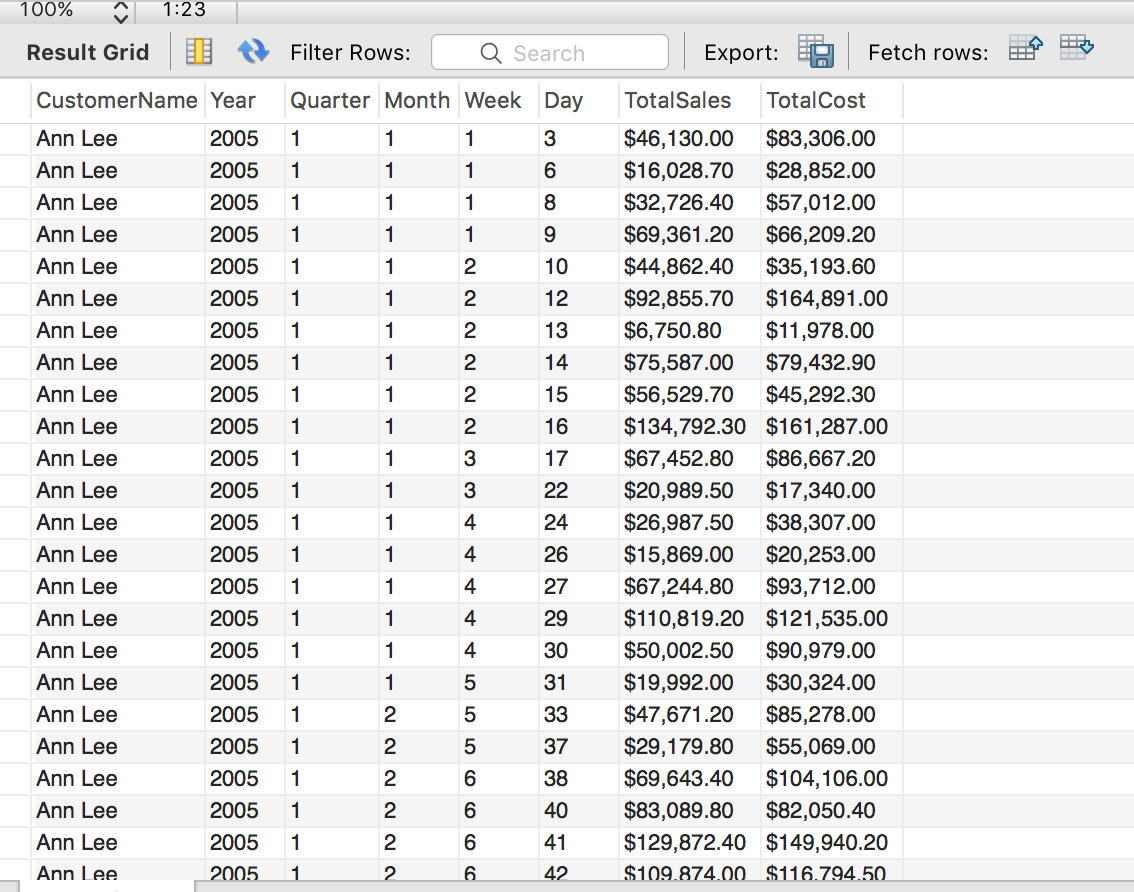
|  |
| --- |
| **User Question/Reporting Need** |
| A similar report showing top customers. |
| **SQL Code** |
| SELECT  c.cutomerName AS CustomerName, sum(sf.salesPrice) AS TotalSales  FROM  Customer c,SalesFact sf  WHERE  c.customerKey = sf.customerKey  group by c.cutomerName  order by sum(sf.salesPrice) desc  limit 20; |
| **Supporting Index(es)** |
|  |

|  |
| --- |
| **User Question/Reporting Need** |
| The number of orders that are not shipped within 10 days of order from PEC. |
| **SQL Code** |
| SELECT  COUNT(Distinct invoiceID) AS OrdersNotShippedwithin10days  FROM  SalesFact  WHERE  division = 'PEC'  AND daystoShip > 10; |
| **Supporting Index(es)** |
|  |

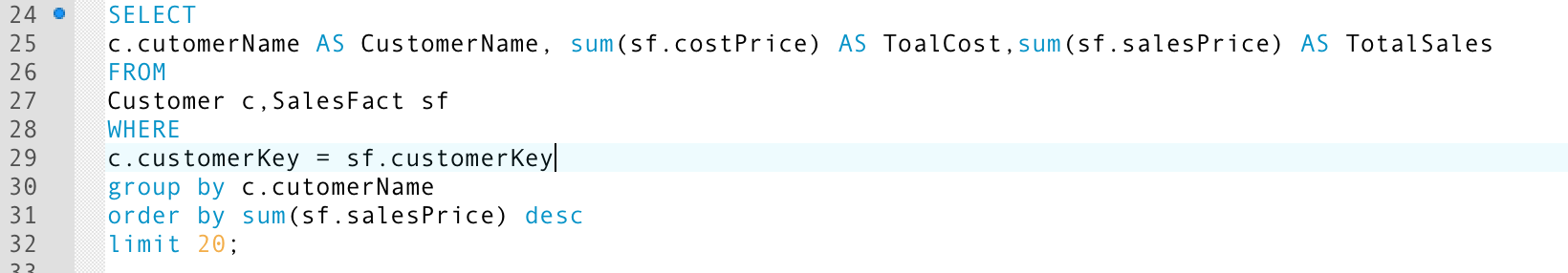
## Views & Summaries

View:

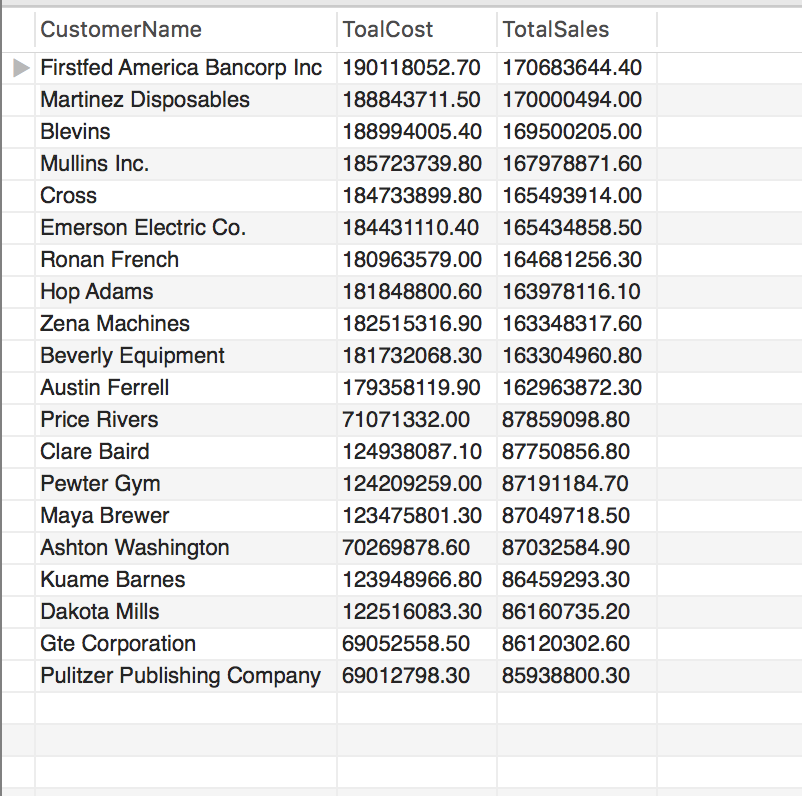
Summary for a report that shows the sales, and costs associated with each customer or customer type on an annual, quarterly, monthly or weekly basis.



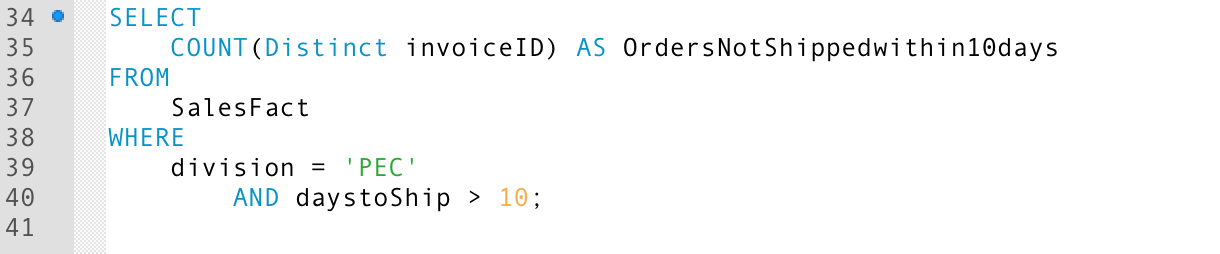
View:



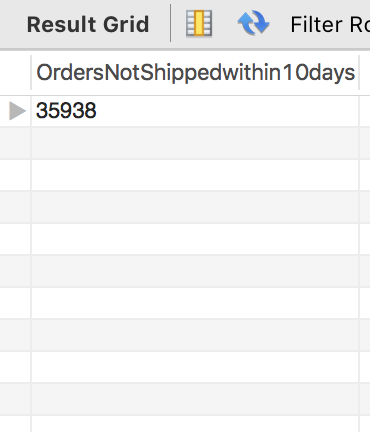
Summary of a similar report showing top 20 customers.



View:



Summary of the number of orders that are not shipped within 10 days of order from PEC.



VIII. Handling Slowly Changing Dimensions (SCD)

We have implement SCD for customer and product table of our FINANCEDW. Following is the summery of the same

Slowly Changing Dimension 1:

Retain Original.

Let us assume that one of our customers have given their zip code which is not correct and PEC does not want to maintain the old zip code, and updates the new one. So we can implement SCD Type 1 for this, where the history data is lost forever.

Implementation of SCD 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| customerKey | customerName | customerAdd1 | customerAdd2 | customerCity | customerState | customerZip |
| 20 | Labor Ready Inc. | Dept #759 | 4618 Days Drive | Waterville | WA | 98855 |
|  |  |  |  |  |  |  |
| 20 | Labor Ready Inc. | Dept #759 | 4618 Days Drive | Waterville | WA | 98858 |

Slowly Changing Dimension 2:

Retains multiple versions of a dimension row.

For example, let us assume that a unit cost of particular product is being changed but TPC wants to keep the old record as well. In such case SCD type 2 can be implemented where you add a new row with current unit cost and a current flag.

Implementing SCD Type 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| productKey | productDescription | productUnitcost | Flag | Effective Date | Expiration Date |
| 25 | Chalmers Polishing Equipment | 223.30 | Old | 03/15/2010 | 06/13/2011 |
| 777 | Chalmers Polishing Equipment | 237.78 | Current | 06/13/2011 | 12/31/9999 |

Slowly Changing Dimension 3:

To track history both before and after a change. For example, consider that the product with product type description needs to be changed to Photo chemicals and BID need to be changed from Miscellaneous to Chemical we implement SCD 3 for this scenario where we can keep track of previous productDescription, previous product BUID and previous productBUIDName along with the current updated details.

Original Records before implementing SCD Type 3:

|  |  |  |  |
| --- | --- | --- | --- |
| productKey | productTypeDescription | productBUID | productBUIDName |
| 3 | Covers | D | Miscellaneous |

After implementing SCD Type 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| productKey | Original  product  TypeDescription | Current  product  TypeDescription | Original  product  BUID | Current  product  BUID | Original product  BUIDName | Current product  BUIDName |
| 3 | Covers | Photo Chemicals | D | C | Miscellaneous | Chemicals |

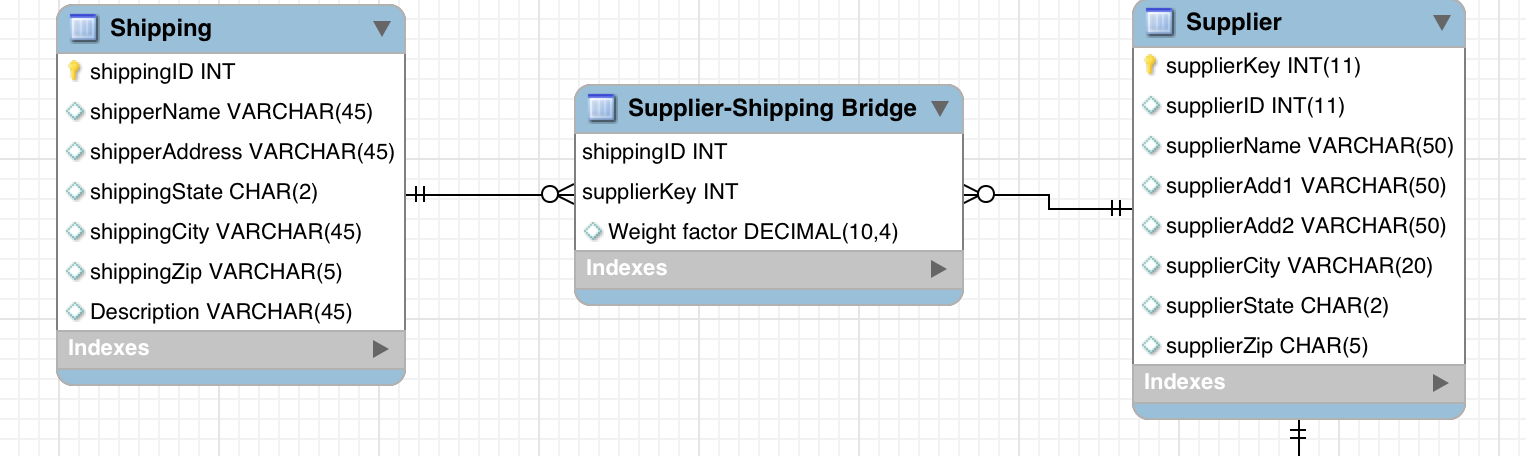
IX Many-to-Many (N-M) Relationship Implementation Option

The implementation strategy we propose for our data warehouse for many-many relationship is the use of the Bridge table method. According to Song, Rowen, Medsker, Ewen (2001), the use of the bridge table is made to connect multiple shipping companies with the supplier. The rationale behind this implementation is that supplier will be associated with many shipping companies and many shipping companies will be associated with many suppliers. The bridge table used here is similar to an intersection table used for many- many relationships between entities. The only difference is the use of weighting factor in a data warehouse as compared to relational database system.

The other options available to take care of many-many relationships are as follows:

* Boolean Column Method
* The Multiple Column Method

The Boolean Column Method contains of creating a column for each possible value. This can consume a lot of space in a problem like stated in the project. If there are a large number of shipping companies, then the fact table will be overloaded with columns. Likewise, the Multiple Column Method consists of columns for the total number of choices one can make. Using this option means having a lot of null values which is not efficient.  Hence we choose Bridge Table Method which is much more efficient than the other two methods and also takes less storage space.



X. REFERENCE:

Hamel, R. (2007, January 19). Implementing many-to-many relationships in data warehousing. Retrieved from Pythian website: http://www.pythian.com/blog/ implementing-many-to-many-relationships-in-data-warehousing/

Kimball, R., Ross, M., & Thornthwaite, W. (2011). Many- valued dimensions with bridge table [Introduction to Dimensional Modelling]. In Data warehouse lifecycle toolkit (Volume 2 ed., pp. 270-273). John Wiley & Sons.

Song, Y., Rowen, W., Medsker, C., & Ewen, E. (2001). An analysis of many-to-many relationships between fact and dimension tables in dimensional modeling. International Workshop on Design and Management of Data Warehouses, 39, pp. 6-1-6-13.