***The Product Company***

**~ Final Data Mart Development Report ~**

Team # \_\_\_\_\_\_1\_\_\_\_\_\_\_\_

Team Members

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Edward Rao\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Chun-Yu Lee\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Abhinav Srinivasan\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Parvati Neeralgi\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

December 15, 2021

ISTE-DW Data Warehousing

~ Table of Contents ~

[I. Data Mart Design Definition 2](#_30j0zll)

[1. Universe of Discourse 2](#_1fob9te)

[2. Information Package 2](#_3znysh7)-3

3. Entity Definitions 3-9

[II. Dimensional Model](#_2et92p0) 10

[III. Data Staging: ETL – Data Extract File Definitions](#_tyjcwt) 11

[IV. Data Staging: ETL – Source-to-Target Mappings](#_3dy6vkm) 12

[V. SQL Code – Tables & Constraints](#_1t3h5sf) 12

[VI. Data Staging Activities - ETL](#_4d34og8) 13-19

[1. Data Cleansing](#_2s8eyo1) 13-14

[2. Data Transformation](#_17dp8vu) 15-19

[3. Table Population](#_3rdcrjn) 19

[VII. End User Applications](#_26in1rg) 20-29

[1. SQL99 Queries](#_lnxbz9) 20-23

[2. A Views](#_35nkun2) 23-24

3. Aggregated Data Marts …….……………………………………………........25-29

VIII. Handling Slowly Changing Dimensions (SCD)……..…………… ….……. 30-33

IX. N-M Implementation Option…………………………………………………34-35

X Appendix (Fix Lab #3)…………………………………………………………..36

# I. Data Mart Design Definition

## 1. Universe of Discourse

|  |
| --- |
| The universe of discourse for this data mart is to identify, collect, and manage the historical financial performance for product sales in the TPC (Product Company) and its three divisions of TPC-E, TPC-W, and PEC through customer, product, supplier, and date information in order to optimize sales and minimize costs. |

## 2. Information Package

Process Name: \_\_\_\_\_\_Financial Analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Grain: \_\_\_\_\_\_The sales invoice per product\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

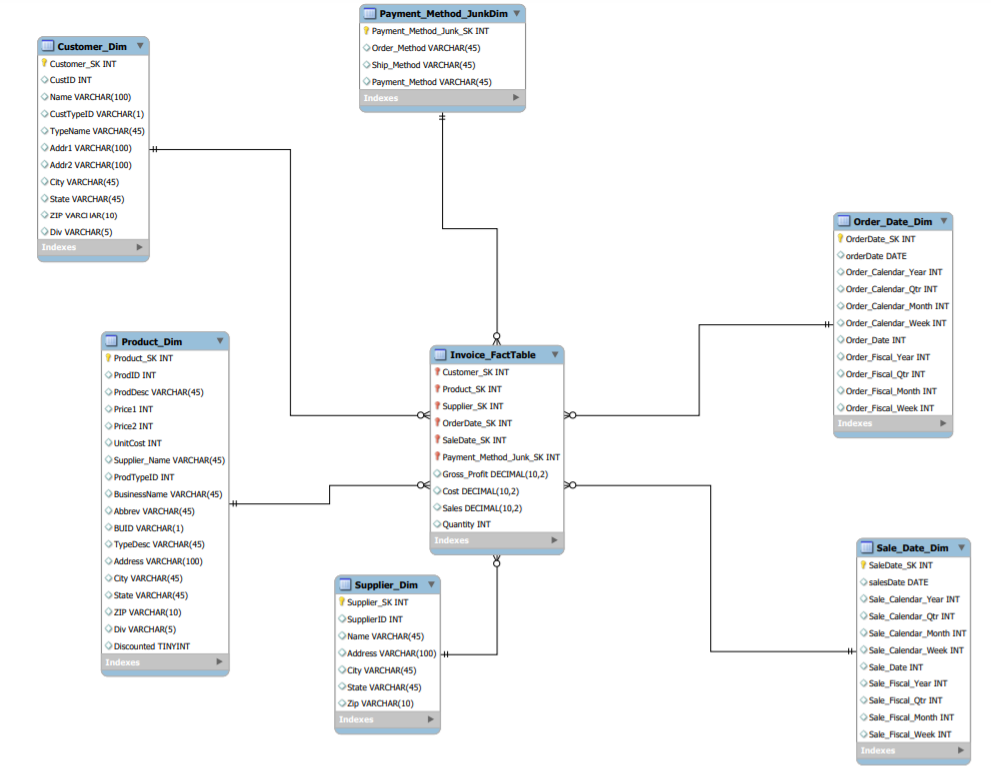
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Customer | Product | Supplier | Sale\_Date | Order\_Date | Payment\_Method\_Junk |
| Customer\_Name | Product\_ID | Supplier\_Name | Sale\_Date | Order\_Date | Order\_Method |
| Customer\_Address | Product\_Description | Supplier\_Address | Sale\_Calendar\_Year | Order\_Calendar\_Year | Ship\_Method |
| Customer\_City | Price1 | Supplier\_City | Sale\_Calendar\_Quarter | Order\_Calendar\_Quarter | Payment\_Method |
| Customer\_State | Price2 | Supplier\_State | Sale\_Calendar\_Month | Order\_Calendar\_Month | Discounted |
| Customer\_Zip | Unit\_Cost | Supplier\_Zip | Sale\_Calendar\_Week | Order\_Calendar\_Week |  |
| Customer\_Type\_ID | Supplier\_Name |  | Sale\_Fiscal\_Year | Order\_Fiscal\_Year |  |
| Customer\_Type\_Name | Product\_Type\_ID |  | Sale\_Fiscal\_Quarter | Order\_Fiscal\_Quarter |  |
| Division | Product\_Type\_Desc |  | Sale\_Fiscal\_Month | Order\_Fiscal\_Month |  |
|  | Business\_Unit |  | Sale\_Fiscal\_Week | Order\_Fiscal\_Week |  |
|  | Business\_Unit\_Name |  |  |  |  |
|  | Business\_Unit\_Abbrev |  |  |  |  |
|  | Division |  |  |  |  |

Facts: \_\_\_\_\_\_\_Cost, Sales, Quantity, Gross profit\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Entity Definitions

|  |  |
| --- | --- |
| **Entity** | **Entity Definition** (*genus differentia*) |
| Customer\_dim | Customer\_SK  Description: The surrogate key of the dimension table to identify the customer  Type: Integer  CustID  Description: A unique identification number to identify the customer in the table.  Type: Integer    Name  Description: The name of the customer in the table.  Type: Varchar    Cust TypeID  Description: A unique identification letter to identify the customer type in the table  Type: Varchar    TypeName  Description: Indicates the name of the customer type in the table.  Type: Varchar  Addr1 Description: Address line 1 of the customer present in the dimensional table. Type: Varchar  Addr2  Description: Address line 2 of the customer present in the dimensional table. Type: Varchar  City  Description: Indicates the city of the customer in the dimensional table.  Type: Varchar    State Description: Indicates the state of the customer present in the dimensional table in the form of its 2 letter abbreviation Type: Varchar  ZIP Description: Indicates the zip code of the customer. Type: Varchar    Div Description: It indicates the division that the customer is involved with  Type: Varchar |
| Invoice\_fact | Customer\_SK  Description: The primary key of the customer dimensional table used in invoice fact table to retrieve the customer attributes.  Type: Integer    Product\_SK  Description: The primary key of product dimensional table used in invoice fact table to retrieve the product details attributes.  Type: Integer  Supplier\_SK  Description: The primary key of supplier dimensional table used in invoice fact table to retrieve the supplier details attributes.  Type: Integer    OrderDate\_SK  Description: The primary key of order date dimensional table used in invoice fact table to retrieve the order date details attributes.  Type: Integer  SaleDate\_SK  Description: The primary key of the sale date dimensional table used in invoice fact table to retrieve the details of sale date attributes.  Type: Integer    Payment\_Method\_Junk\_SK  Description: The primary key of the payment method junk dimensional table used in invoice fact table to retrieve the details of junk attributes.  Type: Integer  Gross\_Profit  Description: The profit calculated .  Type: Decimal  Cost  Description: The cost of the product  Type: Decimal.  Sales  Description: The total sales of products.  Type: Decimal    Quantity  Description: The quantity of products sold.  Type: Integer |
| Order\_date\_dim | OrderDate\_SK  Description: The surrogate key to identify the order date in the dimensional table  Type: Integer    orderDate  Description: It indicates the date on which product is ordered.  Type: Date  Order\_Calendar\_Year  Description: It indicates the year in product order date.  Type: Integer    Order\_Calendar\_Qtr  Description: It indicates the quarter in product order date.  Type: Integer  Order\_Calendar\_Month  Description: It indicates the month in product order date.  Type: Integer    Order\_Calendar\_Week  Description: It indicates the week in which product is ordered.  Type: Integer  Order\_Fiscal\_Year  Description: It indicates the financial year of order.  Type: Integer    Order\_Fiscal\_Qtr  Description: It indicates the financial quarter of order.  Type: Integer  Order\_Fiscal\_Month  Description: It indicates the financial month of order.  Type: Integer    Order\_Financial\_Week  Description: It indicates the financial week in which product is ordered.  Type: Integer |
| Payment\_method\_junkdim | Payment\_Method\_Junk\_SK  Description: The surrogate key used to identify payment methods in the junk dimension table.  Type: Integer    Order\_Method  Description : Indicates the method used to order a product.  Type: Varchar  Ship\_Method  Description: Indicates the method used to ship a product.  Type: Varchar    Payment\_Method  Description: Indicates the method used to make payment for ordering a product.  Type: Varchar |
| Product\_dim | Product\_SK  Description: Indicates a surrogate key to identify product in table  Type: Integer    ProdID  Description: A unique identification number used to identify the product in a dimensional table.  Type: Integer  ProdDesc  Description: It describes the characters of the product  Type : Varchar    Price1  Description: It indicates the initial price of the product  Type : Integer  Price2  Description: It indicates the Final price of the product  Type : Integer    UnitCost:  Description: It indicates the cost of the product unit.  Type: Integer  ProdTypeID  Description: It is unique identification number used to identify the type of product  Type: Integer    Business Name  Description: It indicates the name of the business organization  Type: Varchar  Abbrev  Description: It indicates the abbreviation used for the product  Type: Varchar    BUID  Description : A unique identification value used to indicate business unit  Type: Varchar  TypeDesc  Description: It gives description about product type.  Type: Varchar    Address  Description: It indicates the address of the product stored.  Type: Varchar  City  Description: It indicates the city of the product stored.  Type: Varchar    State  Description: It indicates the state of the product stored.  Type: Varchar  ZIP  Description: It indicates the zip code of the product stored.  Type: Varchar  Div  Description: It indicates division of the product stored.  Type: Varchar    Discounted  Description: It indicates the discount given on purchase of product  Type: TINYINT |
| Sale\_date\_dim | SaleDate\_SK  Description: The surrogate key used to identify the sales date in the dimensional table  Type: Integer    salesDate  Description: It indicates the date on which product is sold.  Type: Date  Sale\_Calendar\_Year  Description: It indicates the year in product sales date.  Type: Integer    Sale\_Calendar\_Qtr  Description: It indicates the quarter in product sales date.  Type: Integer  Sale\_Calendar\_Month  Description: It indicates the month in product sales date.  Type: Integer    Sale\_Calendar\_Week  Description: It indicates the week in which product is sold.  Type: Integer  Sale\_FIscal\_Year  Description: It indicates the financial year of product sold.  Type: Integer    Sale\_Fiscal\_Qtr  Description: It indicates the financial quarter of sales.  Type: Integer  Sale\_Fiscal\_Month  Description: It indicates the financial month of sales.  Type: Integer  Sale\_Financial\_Week  Description: It indicates the financial week in which product is sold.  Type: Integer |
| Supplier\_dim | Supplier\_SK  Description: The surrogate key is used to identify supplier in dimensional table  Type: Integer    SupplierID  Description: It is a unique identification number of supplier  Type: Integer  Name  Description: It indicates the name of the supplier  Type: Varchar    Address  Description: It gives the address of the supplier  Type : Varchar  City  Description: It gives the city of the supplier  Type: Varchar    State  Description: It gives the state of the supplier  Type: Varchar  Zip  Description: It gives the zip code of the supplier  Type: Varchar |

# II. Dimensional Model



III. Data Staging: ETL – Data Extract File Definitions

# 

# The following are the data files included in this report:

# 

|  |  |
| --- | --- |
| File Name | Data Format |
| customer.csv (TPCE) | CSV, semicolon-delineated, quotes |
| customer\_type.csv (TPCE) | CSV, semicolon-delineated, quotes |
| product.csv (TPCE) | CSV, semicolon-delineated, quotes |
| prod\_type.csv (TPCE) | CSV, semicolon-delineated, quotes |
| business\_unit.csv (TPCE) | CSV, semicolon-delineated, quotes |
| supplier.csv (TPCE) | CSV, semicolon-delineated, quotes |
| invoice.csv (TPCE) | CSV, comma-delineated |
| invoice\_details.csv (TPCE) | CSV, comma-delineated |
| TPCWcustomer.csv (TPCW) | CSV, semicolon-delineated, quotes |
| TPCWcustomer\_type.csv (TPCW) | CSV, semicolon-delineated, quotes |
| TPCWproduct.csv (TPCW) | CSV, semicolon-delineated, quotes |
| TPCWprodcut\_type.csv (TPCW) | CSV, semicolon-delineated, quotes |
| TPCWbusiness\_unit.csv (TPCW) | CSV, semicolon-delineated, quotes |
| TPCWinvoice.csv (TPCW) | CSV, comma-delineated |
| PECcustomer.csv (PEC) | CSV, semicolon-delineated |
| PECcustomer\_type.csv (PEC) | CSV, semicolon-delineated, quotes |
| PECproduct.csv (PEC) | CSV, semicolon-delineated |
| PECproduct\_type.csv (PEC) | CSV, semicolon-delineated, quotes |
| PECbusiness\_unit.csv (PEC) | CSV, semicolon-delineated, quotes |
| PECmanufacturingCosts.csv (PEC) | CSV, vertical bar |
| PECinvoice.csv (PEC) | CSV, comma-delineated |

# 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.

# The source-to-mapping table can be found attached in the folder titled “ETL Source to Target”. An image is included below

Table

Description automatically generated

# V. SQL Code – Tables & Constraints

# Files (dump file and schema) included in the folder. An image is included below.

Text

Description automatically generated

# VI. Data Staging Activities - ETL

## 1. Data Cleansing

|  |  |  |  |
| --- | --- | --- | --- |
| **DM Table** | **Attribute** | **Problem** | **Resolution Strategy** (attach code) |
| TPCW\_Customer | All | “(quotes) are present before and after the value | Replace in string “ quotes |
|  | Address | Few entries have Dept# or Suite# | Modified JavaScript for splits and replace string if needed |
|  | Address | Two fields are required for Address | Split into Address1 and Address 2 |
|  | DIV | Missing field | Value mapper is used to add DIV |
|  | State | Full state name isn’t used | Replace string : eg - CA to California |
| TPCW\_Product | All | Remove quotation marks | Replace in string “ quotes |
|  | City, State | City and state are clubbed | Split using delimiter into separate fields |
|  | Product | Duplicate entries | Removed duplicate values |
| TPCW\_Invoice | All | Except ID and other null attributes | Null if the value is empty |
|  | CustID | Negative ID | Modified JavaScript: use absolute value |
|  | salesDate | Date format | Changed into MM/DD/YY |
|  | InvoiceID | Record with only ID | Removed entry by filtering |
| TPCE\_Customer | Name | Abbreviations are used | Replace with the root word |
|  | DIV | Missing field | Value mapper is used to add DIV |
| TPCE\_Product | All | “(quotes) are present before and after the value | Replace in string “ with white spaces |
|  | Supplier\_Name,  Address, City, Zip | Missing fields for supplier | Stream lookup and add using SupplierID(TCPE) and Supplier Dimension |
|  | Saleby | Missing fields | Add fields from TPCE |
|  | discounted | Missing fields | Add from TPCE\_Product and Invoice Table |
|  | BUID, BusinessName, Abbrev | Missing fields | Add by using businessuni to merge |
| TPCE\_Invoice | salesDate | Date format | Changed into MM/DD/YY |
| PEC\_Customer | All | “(quotes) are present before and after the value | Replace in string “ quotes |
|  | Address | Two fields are required for Address | Split into Address1 and Address 2 |
|  | Address1, Address2 | Abbreviations are used | Replacing with root word  (St : Street) |
|  | Name | Abbreviations are used | Replacing with root word  (Inc : Incorporated) |
| PEC\_Product | orderDate | Date format | Changed into MM/DD/YY |
|  | salesDate | Date format | Changed into MM/DD/YY |
|  | unit cost | unit cost is calculated using manufacturing costs | Calculator in Pentaho; manufacturing costs/qty |
| PEC\_Invoice | Sales\_date | Date format | Replace in string “ with white spaces and change format to MM/DD/YY |
|  | Order\_date | Date format | Replace in string “ with white spaces and change format to MM/DD/YY |
|  | shipping\_method | Bad value | Replaced |
|  | price | Incorrect value | Matched from TCPE\_Invoice |

## 2. Data Transformation

|  |  |
| --- | --- |
| **DM Table** | **Image Creation Process** (attach code) |
| Customer\_dim | TPCW\_Customer\_Input   * Commas in the address of the customer were replaced by using replace in string * Set the size of ZIP to 4 using javascript   characters ' ˆ/s were replaced using regex   * 1977 Symbiotic Turnpike,Dept #805 replaced by Dept #805 1977 Symbiotic Turnpike using regex (these are all just address formatting for TPCW to split later) * 3072 Cursus Avenue,Suite 30 replaced by Suite 303072 Cursus Ave using regex * 11533 Wonderingly Drive,Suite 50 replaced by Suite 50 11533 Wonderingly Drive using regex * 3085 Breakthrough Drive,Dept #80 replaced by Dept #80 3085 Breakthrough Drive using regex * 7196 Heaver Lane,Dept #555 replaced by Dept #555 7196 Heaver Lane using regex * District of Columbia replaced by DC * Florida replaced by Fl * Split the address field into address1 and address 2 with comma delimiter. * adding division field of type string to customer using add constants * Mapping div using value mapper   Standardize TPCW by replacing some fields   * (Inc.\*) replaced with Incorporated in name field using regex * (Corp[\s\.]|Corporation|Corp) replaced with Corporation in name field using regex * (\bCo\b) replaced with Company in name field using regex * St replaced with Street in address field * Ave replaced with Avenue in address field * Rd replaced with Road in address field * (^C.\*) replaced with C using regex in custtype field (All fields with C) * (^E.\*) replaced with E using regex in custtype field * (^G.\*) replaced with F using regex in custtype field (G for Govt, F for Federal) * (^S.\*) replaced with S using regex in custtype field * All the states names were renamed using modified java script * Some field names were renamed using select values   PEC\_Customer\_Input   * Standardize PEC using replace in value * (Inc.\*) replaced with Incorporated in name field using regex * (Corp[\s\.]|Corporation|Corp) replaced with Corporation in name field using regex * (\bCo\b) replaced with Company in name field using regex * St replaced with Street in address field * Ave replaced with Avenue in address field * Rd replaced with Road in address field * (^C.\*) replaced with C using regex in custtype field * (^E.\*) replaced with E using regex in custtype field * (^G.\*) replaced with F using regex in custtype field * (^U.\*) replaced with F using regex in custtype field (U for US government) * (^S.\*) replaced with S using regex in custtype field * Add Div to PEC using add constants * Mapping Div to PEC using vale mapper * Split the address into two fields * Few fields were renamed using select values * Rows are sorted by Name   TPCE\_Customer\_Input   * "" were removed using replace in string * Field names were renamed using select values * Standardize TPCE using replace in value * (Inc.\*) replaced with Incorporated in name field using regex * (Corp[\s\.]|Corporation|Corp) replaced with Corporation in name field using regex * (\bCo\b) replaced with Company in name field using regex * St replaced with Street in address field * Ave replaced with Avenue in address field * Rd replaced with Road in address field * (^C.\*) replaced with C using regex in custtype field * (^E.\*) replaced with E using regex in custtype field * (^G.\*) replaced with F using regex in custtype field * (^U.\*) replaced with F using regex in custtype field * (^S.\*) replaced with S using regex in custtype field * Add Div to TPCE using add constants * Mapping Div to TPCE using vale mapper   Address was aligned using string operations.  TPCE\_CustType\_Input   * "" were removed using replace in string * Field names were renamed using select values * used append streams operation |
| Product\_dim | PEC\_Product\_Input   * " is removed using replace in string * unit cost is calculated using calculator in pentaho * PEC\_Manufacturing\_cost * Year value is modified. * Rows are sorted by year, month, prodID * Grouped by year, month, prodID * Few field names are renamed.   Cleaned PEC Invoice   * Unit cost was calculated * Rows are sorted by year, month, prodID   Business\_Unit\_Input   * " is removed using replace in string * Few field names are renamed. * Misc Abbr added using java script   TPCW\_ProdType\_Input   * " is removed using replace in string * Few field names are renamed.   TPCW\_Product\_Input   * " is removed using replace in string * Equip was replaced with Equipment using replace in string * Rows were sorted by Prod\_Name * Unique rows were selected by Prod\_Name and duplicate rows were removed * address field was splitted and concatenated * Few field names are renamed. * Standardize TPCW using replace in value * (Inc.\*) replaced with Incorporated in name field using regex * (Corp[\s\.]|Corporation|Corp) replaced with Corporation in name field using regex * (\bCo\b) replaced with Company in name field using regex * St replaced with Street in address field * Ave replaced with Avenue in address field * Rd replaced with Road in address field * Duplicate rows are deleted. * Saleby of type string was added to TPCW   TPCE\_Product\_Input   * " is removed using replace in string * Few field names are renamed. * Duplicate rows are delete * Address field was edited   For each product table, we also join the cleaned columns with discounted from the Invoice data before exporting.  Duplicate rows are removed and PEC, TPCE and TPCW are merged |
| Supplier\_dm | CSV file input Supplier  ext file input : TPCE, PEC   * Replace is used to remove attn: in address 1 and 2 * concatenating address1 and address 2 into address * TPCE Address are concatenated and then rename the rest of the fields * Filtering rows that are PEC * Appending streams supplier and PEC and then with TPCE and above * Supplier IDs are normalized(inc to incorporated) * Check for duplicates * Load files into supplier |
| Date\_dim | PEC\_Invoice\_Input   * Order date and saledate were selected using select values * Date is changed to string and normalized to MM/DD/YY * It is converted to Date type * Year, quarter, month, week, date is used * Fiscal values are calculated using js file * Add surrogate key * Add data into saleDate   InvoiceTPCW\_Input   * Date is changed to string and normalized to MM/DD/YY * It is converted to Date type * Year, quarter, month, week, date is used * Fiscal values are calculated using js file   InvoiceTPCE\_Input   * Date is changed to string and normalized to MM/DD/YY * It is converted to Date type * Year, quarter, month, week, date is used * Fiscal values are calculated using js file |
| invoice\_fact | Text file input : PEC\_Invoice and TCPW\_Invoice   * 2 entries from PEC\_Invoice is deleted * Date format is fixed to MM/DD/YY * Records for shipping and dates are fixed * Rows that are not required are removed * Stream lookup of Price and unit cost is accessed from PEC\_Product and cleansed file is taken * Product IDs are filtered depending on their useability * Gross profit and total sales is calculated * Invoice details are updated into TCPE\_Invoice * Price and unit cost is added/matched from TCPE\_Invoice * Values are rearranged and converted * Sort on invoice id, merge all invoices and provide a combined output |
| Junk\_dim | PEC Invoice Data   * Shipmethod, payment method, and order method are the attributes taken into consideration since they have low cardinality * After standardizing each attribute they are sorted * Duplicate rows are taken care off * Add NA to payment method and order method and appending null values * Joining the rows and creating a text output junk file |

## 3. Table Population

For our table loading, we used Pentaho Kettle’s built-in table output transformation, with the specific steps for each differing slightly based on necessity. These are all included in the folder.

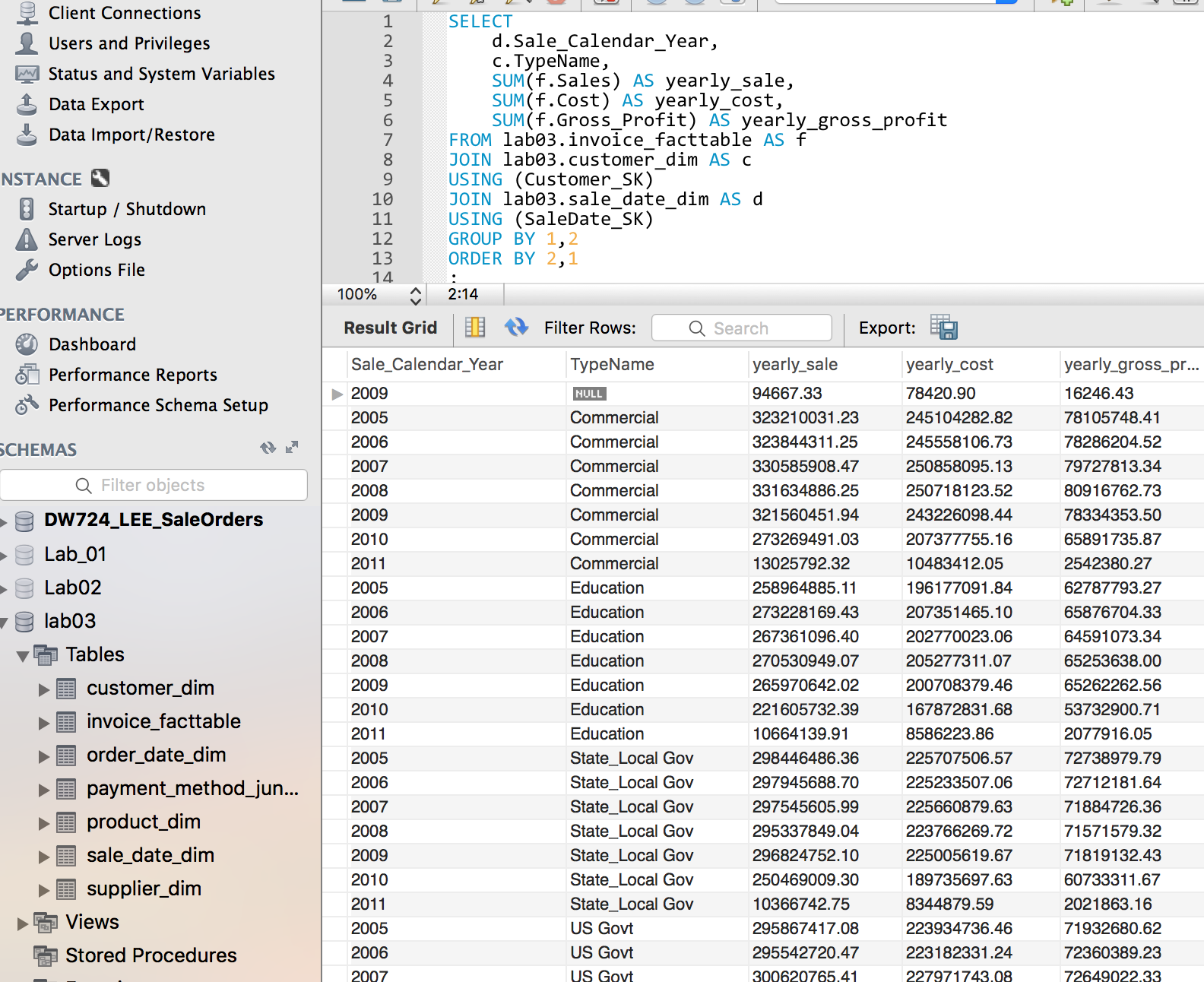
|  |  |
| --- | --- |
| **DM Table** | **Table Population Process** (attach code) |
| Fact Table and Invoice Data | After obtaining the combined invoice file through the above transformations, we began to merge the file through streams with the individual SKs and then dropped all the columns not needed in the end fact table. We then accounted for null SKs, represented by -1, and ordered the entries, deleting duplicates and completing the fact table via the table load to mySQL. |
| Customer Dimension | Although we don’t have a physical unified Customer output because we don’t need to use it for other parts of the data mart, the Customer are combined, sorted by Name. We then add Customer\_SK (incremental, arbitrary, begin at 10001) and used to load into Fact Table |
| Supplier Dimension | After the data is divided into unique rows and arranged by SupplierID, we added Supplier\_SK (incremental, arbitrary, chose 5000001) and used to load into Fact Table |
| Product Dimension | With all 3 of TPCE, TPCW and PEC cleansed and merged, and the product/sum values already calculated, with the PEC version requiring the Manufacturing csv as stated above,  assigned SK (incremental, arbitrary, we chose 100001) for Fact Table |
| Junk Dimension | Joined the shipping, payment, and order methods and assigned SK (incremental, arbitrary, used 1200001) used to load into Fact Table |
| Sales and Order Date Dimension | Sales Date required TPC divisions as well as PEC, while Order Date only required the latter. Both fiscal versions calculated and loaded into dimensions, SKs (incremental, arbitrary, 4001 and 2001 respectively) created for Fact Table |
|  |  |

# VII. End User Applications

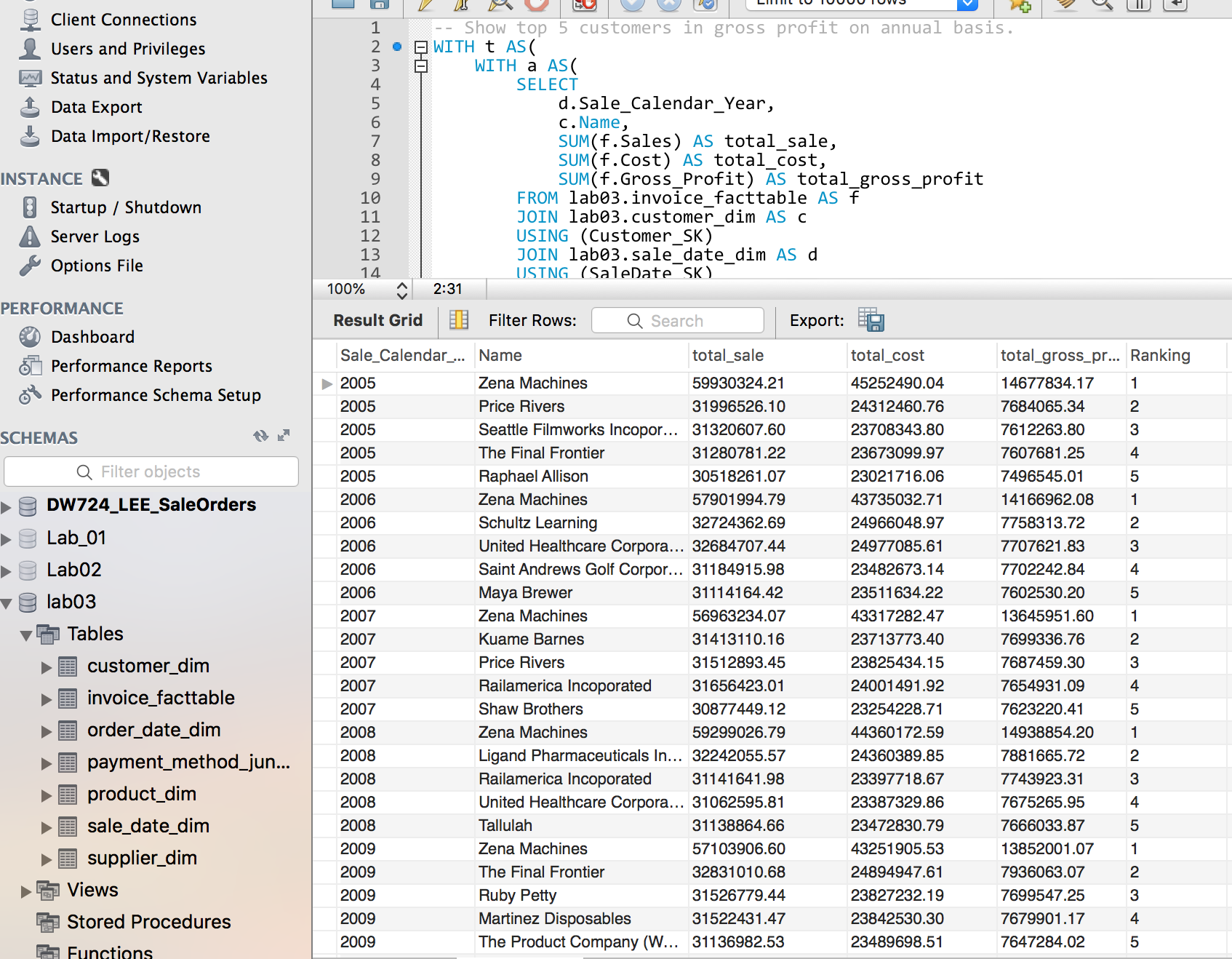
## 

## 1. Queries

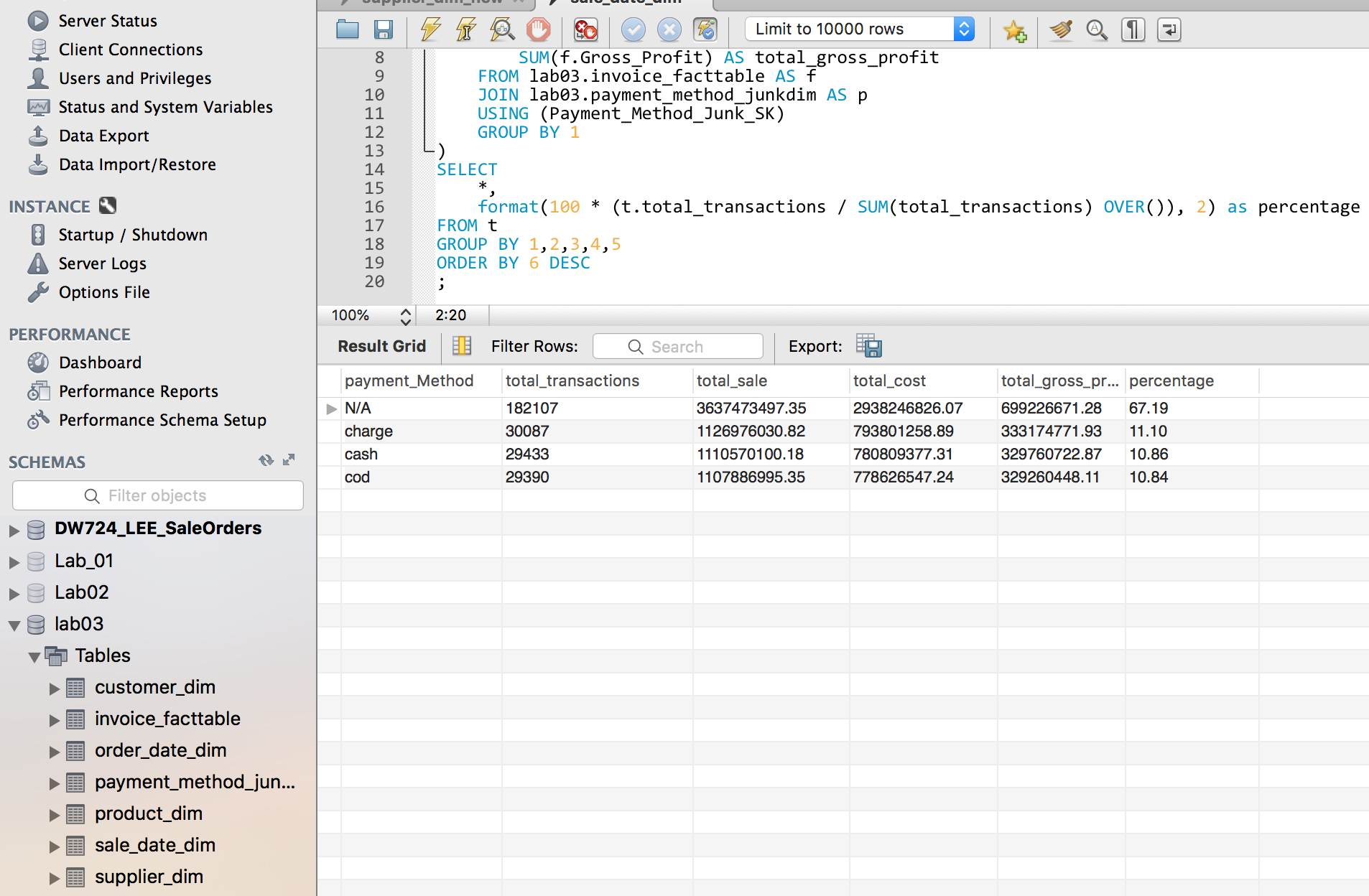
|  |
| --- |
| **User Question/Reporting Need** |
| A report that shows the sales, and costs associated with each customer type on an annual basis. |
| **SQL Code** |
| -- A report that shows the sales, costs, and gross\_profit  -- associated with each customer type on an annual basis  SELECT  d.Sale\_Calendar\_Year,  c.TypeName,  SUM(f.Sales) AS yearly\_sale,  SUM(f.Cost) AS yearly\_cost,  SUM(f.Gross\_Profit) AS yearly\_gross\_profit  FROM lab03.invoice\_facttable AS f  JOIN lab03.customer\_dim AS c  USING (Customer\_SK)  JOIN lab03.sale\_date\_dim AS d  USING (SaleDate\_SK)  GROUP BY 1,2  ORDER BY 2,1  ; |
| **Supporting Index(es)** |
|  |



|  |
| --- |
| **User Question/Reporting Need** |
| Showing top 5 customers in gross profit on an annual basis. |
| **SQL Code** |
| -- Show top 5 customers in gross profit on an annual basis.  WITH t AS(  WITH a AS(  SELECT  d.Sale\_Calendar\_Year,  c.Name,  SUM(f.Sales) AS total\_sale,  SUM(f.Cost) AS total\_cost,  SUM(f.Gross\_Profit) AS total\_gross\_profit  FROM lab03.invoice\_facttable AS f  JOIN lab03.customer\_dim AS c  USING (Customer\_SK)  JOIN lab03.sale\_date\_dim AS d  USING (SaleDate\_SK)  GROUP BY 1,2  ORDER BY 4 DESC  )  SELECT  a.\*,  sum(1) AS Ranking  FROM a  JOIN a as b USING (Sale\_Calendar\_Year)  WHERE a.total\_gross\_profit <= b.total\_gross\_profit  GROUP BY 1,2,3,4,5  )  SELECT  \*  FROM t  WHERE ranking <=5  ORDER BY 1,6  ; |
| **Supporting Index(es)** |
|  |

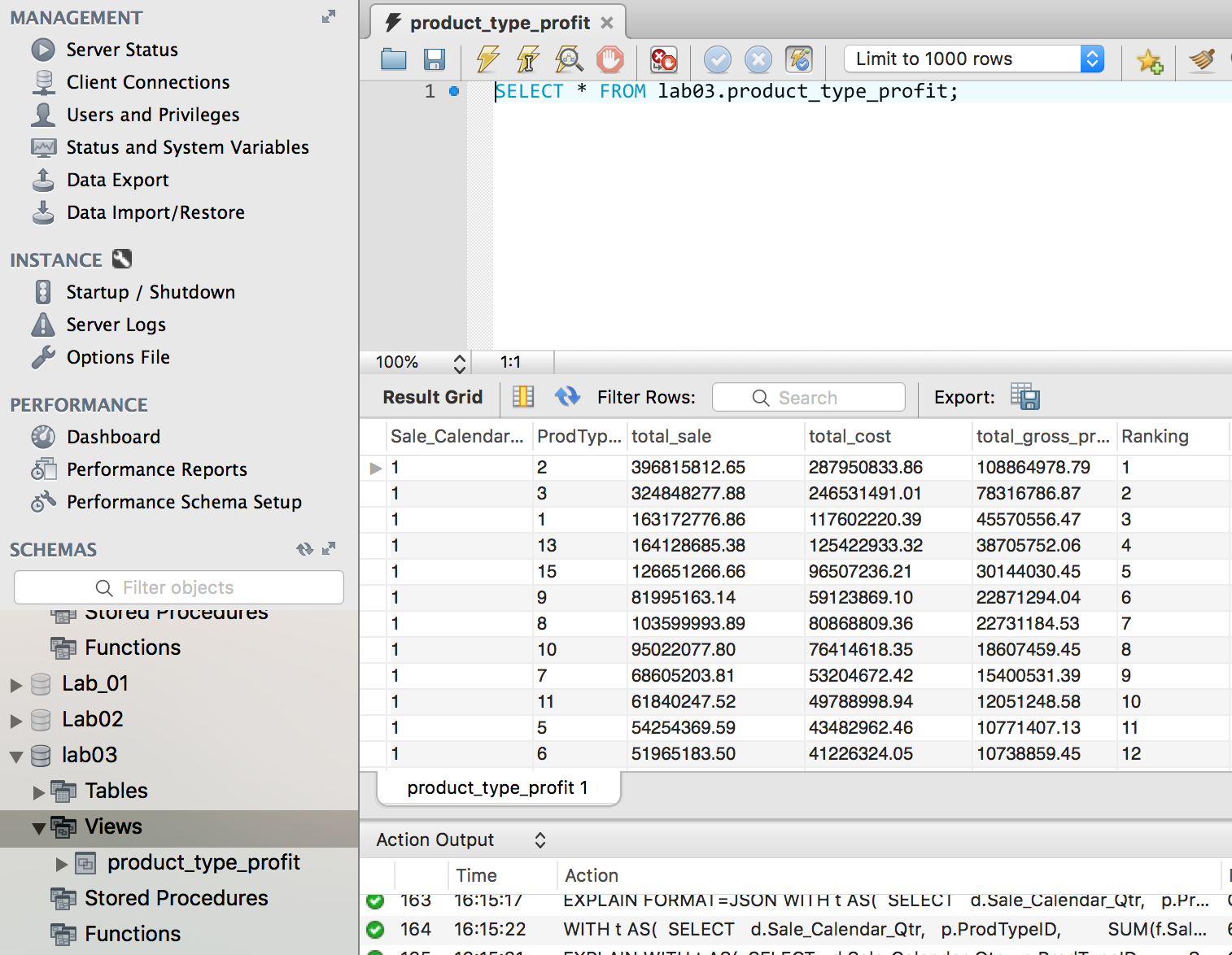


|  |
| --- |
| **User Question/Reporting Need** |
| The percentage of invoices on PEC payment method |
| **SQL Code** |
| -- The percentage of invoices on PEC payment method  WITH t AS(  SELECT  p.payment\_Method,  SUM(1) as total\_transactions,  SUM(f.Sales) AS total\_sale,  SUM(f.Cost) AS total\_cost,  SUM(f.Gross\_Profit) AS total\_gross\_profit  FROM lab03.invoice\_facttable AS f  JOIN lab03.payment\_method\_junkdim AS p  USING (Payment\_Method\_Junk\_SK)  GROUP BY 1  )  SELECT  \*,  format(100 \* (t.total\_transactions / SUM(total\_transactions) OVER()), 2) as percentage  FROM t  GROUP BY 1,2,3,4,5  ORDER BY 6 DESC  ; |
| **Supporting Index(es)** |
|  |



## 2. A View

|  |
| --- |
| **User Question/Reporting Need** |
| Every product type profits on a quarterly basis and the ranking. |
| **SQL Code** |
| -- The product type sales, cost, gross\_profit on quarterly basis  USE `lab03`;  CREATE OR REPLACE VIEW `product\_type\_profit` AS  WITH t AS(  SELECT  d.Sale\_Calendar\_Qtr,  p.ProdTypeID,  SUM(f.Sales) AS total\_sale,  SUM(f.Cost) AS total\_cost,  SUM(f.Gross\_Profit) AS total\_gross\_profit  FROM lab03.invoice\_facttable AS f  JOIN lab03.sale\_date\_dim AS d  USING (SaleDate\_SK)  JOIN lab03.product\_dim AS p  USING (Product\_SK)  GROUP BY 1,2  )  SELECT  t.\*,  sum(1) AS Ranking  FROM t  JOIN t as b USING (Sale\_Calendar\_Qtr)  WHERE t.total\_gross\_profit <= b.total\_gross\_profit  GROUP BY 1,2,3,4,5  ORDER BY 1,6 ASC  ; |

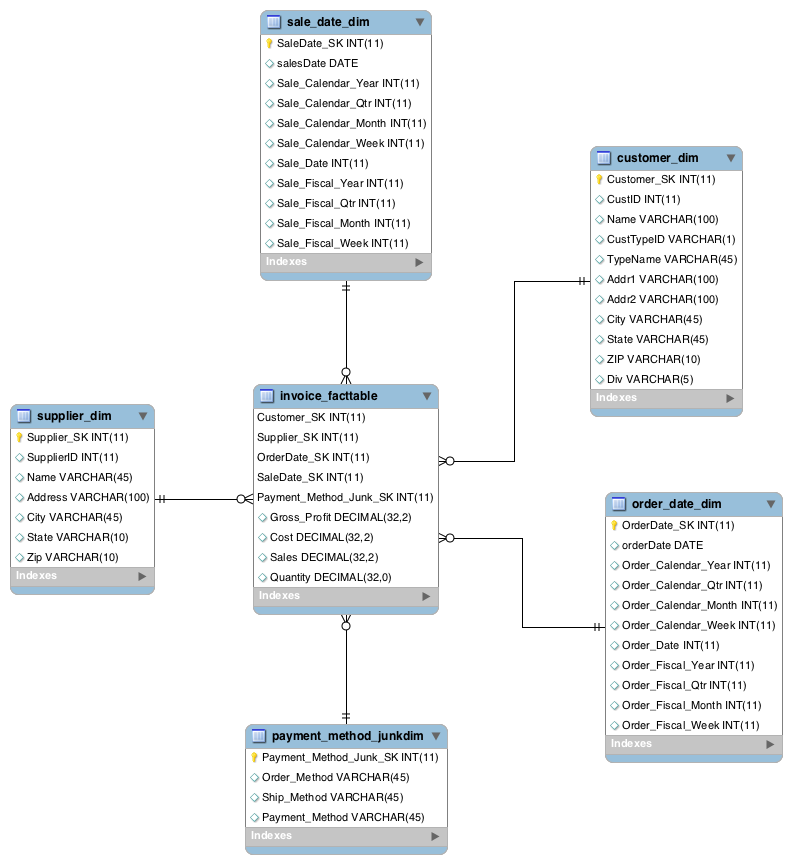


3. Aggregated Data Marts

Aggregation Data Mart (Lost):

In this aggregation data mart, the product dimension is lost.

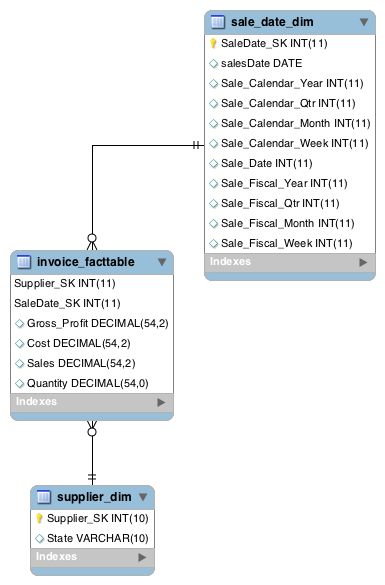
|  |
| --- |
| **Use Case** |
| The user can get the financial result on per transaction.   1. Users can search all the transactions of the company named Firstfed America BanCorporation Incorporated at Jan 2005. 2. Users can search all the transactions of the company located at GA at Jan 2008. |
| **Sample Query 1** |
| SELECT  d.Sale\_Calendar\_Year,  d.Sale\_Calendar\_Month,  c.Name,  f.Cost,  f.Sales,  f.Quantity  FROM lab03\_lost.invoice\_facttable AS f  JOIN lab03\_lost.customer\_dim AS c  USING (Customer\_SK)  JOIN lab03\_lost.sale\_date\_dim AS d  USING (SaleDate\_SK)  WHERE d.Sale\_Calendar\_Year = 2005  AND d.Sale\_Calendar\_Month = 1  AND c.Name = 'Firstfed America BanCorporation Incorporated'  ; |
| **Sample Query 2** |
| SELECT  d.Sale\_Calendar\_Year,  d.Sale\_Calendar\_Month,  c.Name,  f.Cost,  f.Sales,  f.Quantity  FROM lab03\_lost.invoice\_facttable AS f  JOIN lab03\_lost.customer\_dim AS c  USING (Customer\_SK)  JOIN lab03\_lost.sale\_date\_dim AS d  USING (SaleDate\_SK)  WHERE d.Sale\_Calendar\_Year = 2008  AND d.Sale\_Calendar\_Month = 1  AND c.State = 'GA'  ; |



Aggregation Data Mart (Shrunken):

In this aggregation data mart, we keep the Sale Date dimension and shrunken supplier dimension. The grain of the fact table will become suppliers in each state per sale date.

|  |
| --- |
| **Use Case** |
| The user can get the suppliers’ distribution by every state and every sale date.   1. The suppliers in every state delivered the quantity in Jan 2005. 2. The suppliers in CA delivered the quantity in Jan 2008. |
| **Sample Query 1** |
| SELECT  d.Sale\_Calendar\_Year,  d.Sale\_Calendar\_Month,  d.Sale\_Date,  s.state,  f.Cost,  f.Sales,  f.Quantity  FROM lab03\_shr.invoice\_facttable AS f  JOIN lab03\_shr.supplier\_dim AS s  USING (Supplier\_SK)  JOIN lab03\_shr.sale\_date\_dim AS d  USING (SaleDate\_SK)  WHERE d.Sale\_Calendar\_Year = 2005  AND d.Sale\_Calendar\_Month = 1  ; |
| **Sample Query 2** |
| SELECT  d.Sale\_Calendar\_Year,  d.Sale\_Calendar\_Month,  d.Sale\_Date,  s.state,  f.Cost,  f.Sales,  f.Quantity  FROM lab03\_shr.invoice\_facttable AS f  JOIN lab03\_shr.supplier\_dim AS s  USING (Supplier\_SK)  JOIN lab03\_shr.sale\_date\_dim AS d  USING (SaleDate\_SK)  WHERE d.Sale\_Calendar\_Year = 2008  AND d.Sale\_Calendar\_Month = 1  AND s.state = 'CA'  ; |

****

**Aggregation Data Mart (Collapsed**)

In this aggregation data mart, we collapse the dimensions and “merge” them into one new flattened (aka collapsed) table. The aggregation allows the user to quickly focus in on one particular dimension of interest and get overall summarized information

|  |
| --- |
| **Use Case** |
| The user can get the get sales by profit or quantity sold, along with supplementary product information   1. The companies that sold the highest gross profit by state in Qtr 3 of 2009 2. The businesses that sold more than 100 products ranked in all of 2007 |
| **Sample Query 1** |
| SELECT  Name, State, Gross\_Profit, Sale\_Fiscal\_Year, Sale\_Fiscal\_Qtr  FROM collapse.invoice\_facttable  WHERE Name IS NOT NULL  AND Sale\_Fiscal\_Year = 2009  AND Sale\_Fiscal\_Qtr = 3  ORDER BY Gross\_Profit DESC  ; |
| **Sample Query 2** |
| SELECT BusinessName, Sale\_Calendar\_Year, Quantity  FROM collapse.invoice\_facttable  WHERE BusinessName IS NOT NULL  ANS Sale\_Calendar\_Year = 2007  AND Quantity > 100  ORDER BY Quantity DESC  ; |

# 

# VIII. Handling Slowly Changing Dimensions (SCD)

1. For SCD consideration, we will obviously need SCD types besides zero because our data is changing and time-variant. SCD 1 (aka overwriting) is essentially reserved for errors, so would apply only to attributes that hardly, if ever, change, such as an individual customer’s name or the date a transaction was performed, along with identifiers like the customer or product ID.

Besides this, the most commonly used SCD type in this data would be SCD 2, or creating a new dimension row each time the attribute is altered or changed (hence its name SCD). These would be the two different types of SCD besides SCD 0 implemented in the dimensional tables.

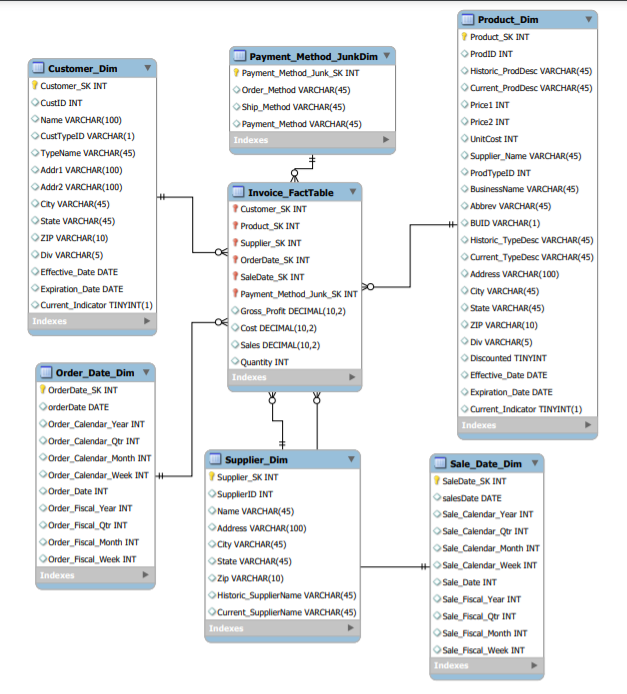
A case can be made for implementing SCD 3, or adding a new attribute, for the Supplier Dimension, particularly the Name attribute in Supplier. This is because a situation could arise where the supplier changes for a particular product, and the manager would like to compare sales like discussed in Lab2. In this case it is very useful for there to be two columns containing the current supplier and previous supplier for a product, and could lead to conclusions being drawn about the performance of each supplier.

For SCD 6, we will be testing on two different items that may be very dynamic, aka Product Description and Type Description. The reasoning behind this is the need to update descriptions and terms often when it comes to sales, and that the owner would likely want to have a complete historical view on the products while also being able to reflect the changes and updates they would require (at least, more so compared to the other attributes).

SCD 6 will be implemented alongside 3 in Supplier Dimension and 2 in Customer Dimension.

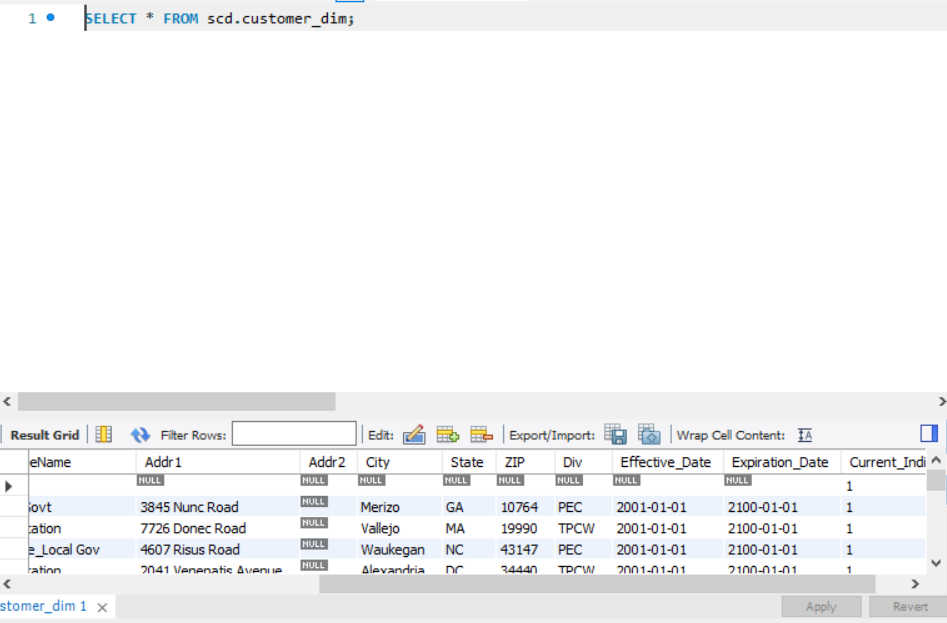
1. We’ll need to create a new SCD model to reflect these implementations, so as not to overwrite our existing data mart. Using a sample of the top 30 entries (except supplier because of number of rows) in customer, supplier, and product (the other rows are SCD 1 so no changes, and also included in folder), we can expect the following:

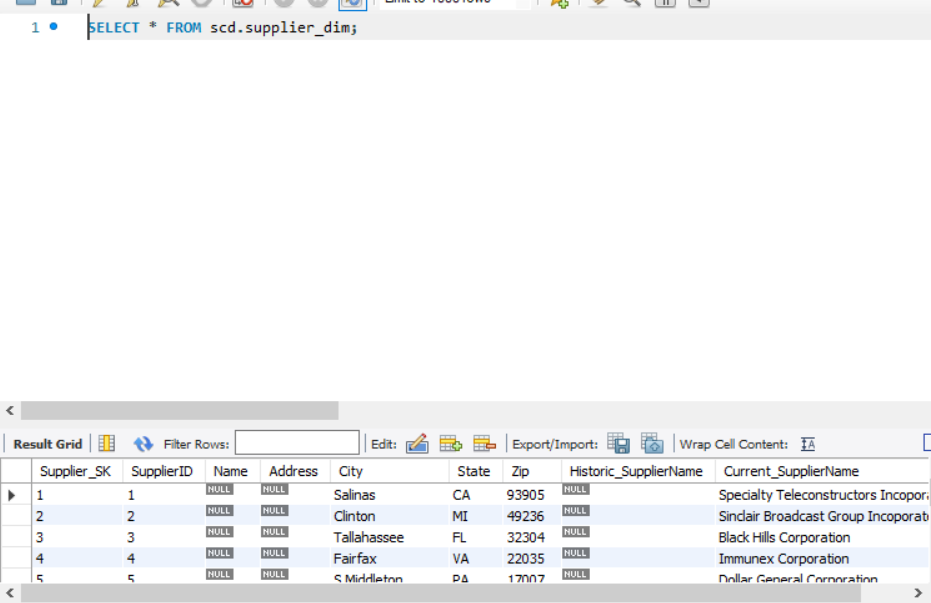
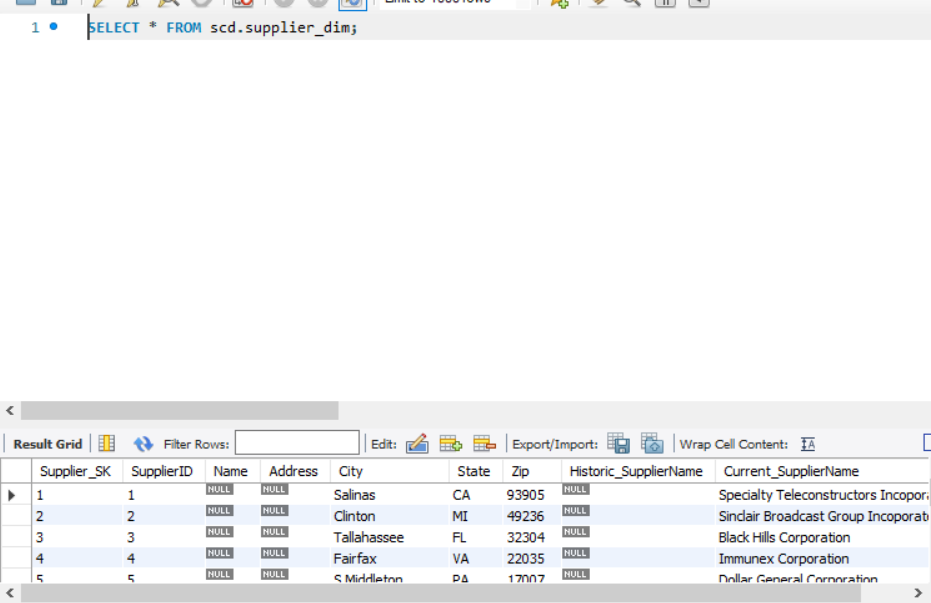
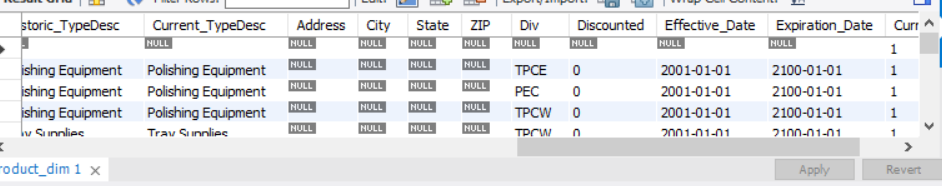
The modified ER diagram:



Changes are made to Product’s Descriptions for SCD 06 while Dates and Current Indicators are also included. For Supplier, a historic and current attribute are added to show SCD03. Both Customer and Product Dimensions also include the effective, expiration date, and indicator attributes for standard SCD 02.

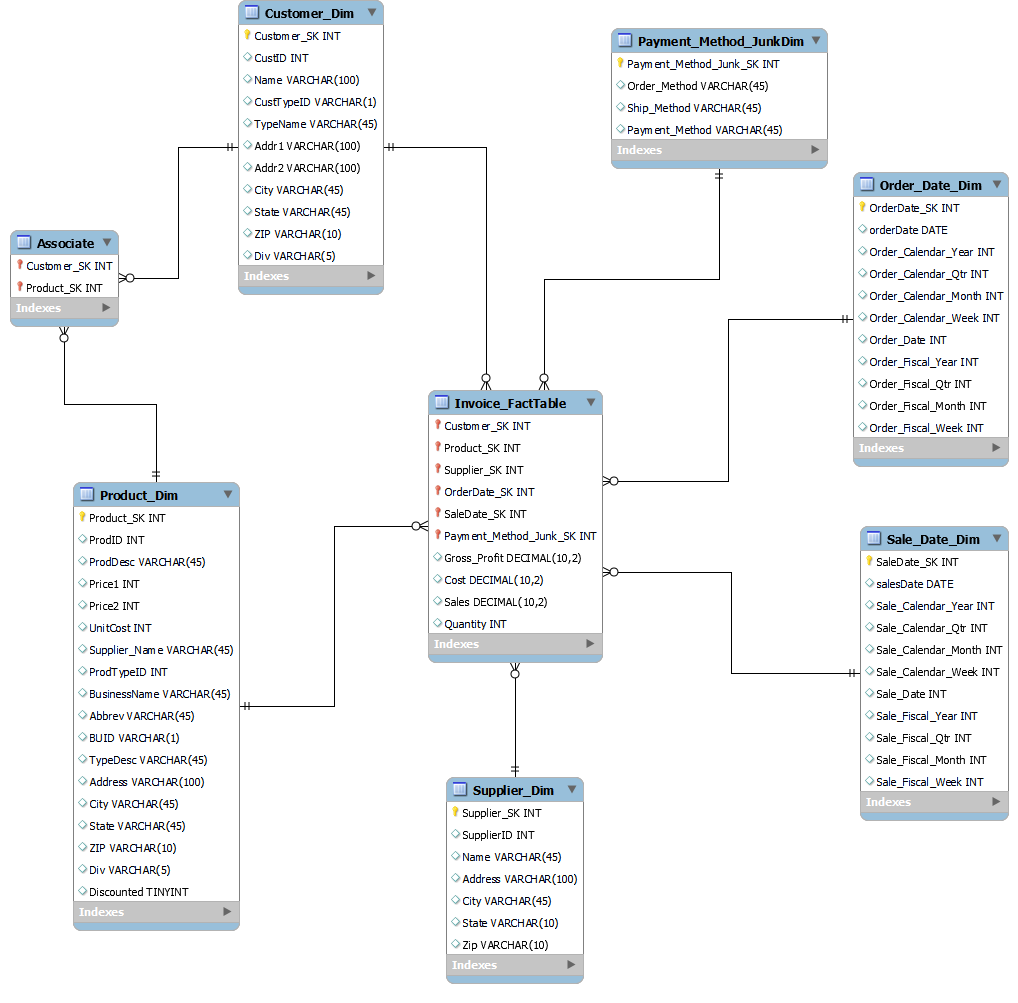
The loaded tables then look as follows:





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

1. A many-to-many relationship refers to a relationship between tables in a database when a parent row in one table contains several child rows in the second table, and vice versa. In relational databases we can not directly implement many to many relations. Consider product and customer , there exist many to many relationships as customers can buy many products and products can be purchased by various customers. Here we can implement many to many relationships by having an associate table in between. The join table has a match field which contains the value of primary keys of the other table it joins.
2. We can also implement many to many relationships by adding the attribute entities to the fact table with the help of surrogate keys.



X. Appendix (Fix Lab #3 Problems)

The following are the problems that we had in our Lab 3 that had to be addressed and fixed in Pentaho.

First, there were still some one-off issues with the data, in particular among the ID values for customer and product along with bad entries that were fixed in Kettle but did not translate well in csv due to spaces or formatting. Here are the cases in particular that were fixed between Lab 3 and the final project:

* PECinvoice ID 12485 formatting and date
* PECinvoice sales amounts, calculated with manufacturing data in Product dimension
* TPCWcustomer zip codes still had only 4 digits; a zero was added but the transformation in Pentaho was incorrect, so it was changed
* TPCWinvoice ID 45461 discounted, ID 3031 Qty missing and fixed
* TPCWinvoice negative CustIDs addressed
* TPCWproduct SK spacing fixed

A major change we had to make was the Address field in PEC and TPCW customer, which had to be split into the “Addr1” and “Addr2” fields to match the TPCE fields and not the other way around.

We also found some cases of Corporation, Company, and Incorporated that we missed and were fixed. This applied to States in TPCW that were not aligned and capitalized, which were addressed.

And finally, because of loading times we weren’t able to fully load the data for Lab3, which was obviously fixed for the final report.