**NAME : s rAMASARMA aNNAPARTHI**

**NETID : VVC150130**

**BUAN 6320.005 – Database Foundations for Business Analytics**

**INDIVIDUAL Project, Fall’18**

**Objective**

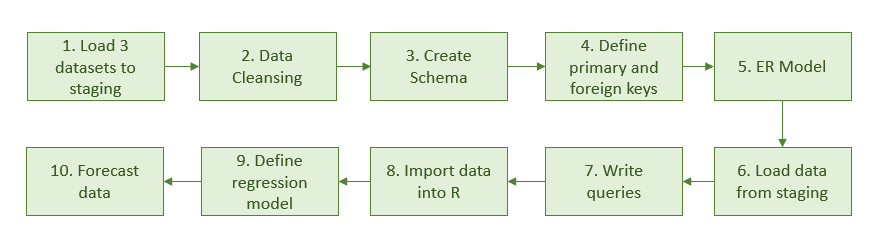
The objective of this project is to extract, transform, normalize and load the raw data of a company into a database model and then write some queries to analyse the data underlying the model. Finally, perform regression analysis to predict forecast of sales for the next year based on few explanatory variables.

**Workflow**

Technologies used to accomplish this project.

|  |  |
| --- | --- |
| Category | Software/Tool |
| Database | MS SQL Server |
| Modelling/Regression | R |
| ETL | SQL Server Import & Export Wizard |

Following are the steps I followed for this project:

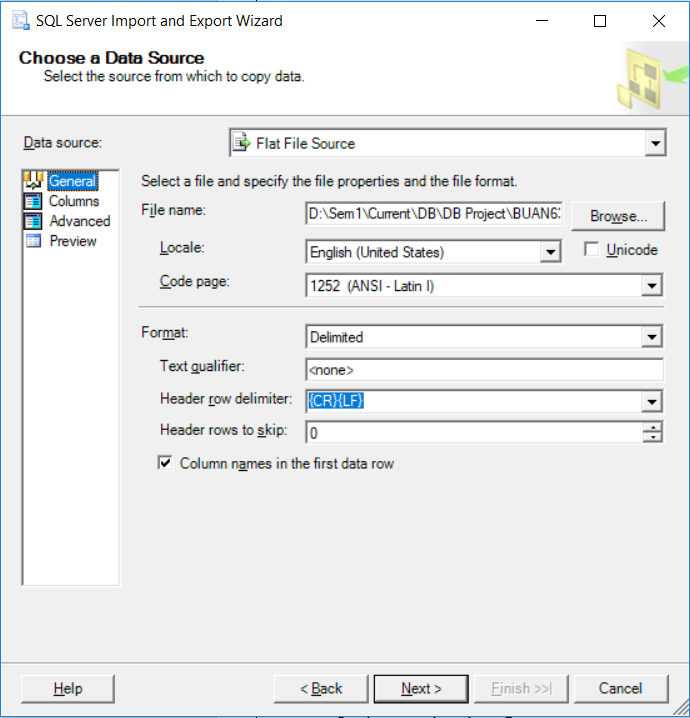


1. **Data load to staging**
2. Created a database, “DbProject”, in MS SQL Server.
3. Loaded the three data sets (CSV data) into the database using SQL Server Import & Export wizard.
4. Following are the issues I faced while doing this step
   1. **Issue 1**: Got an error while loading Dataset3 with the following message: *"Text was truncated or one or more characters had no match in the target code page."*

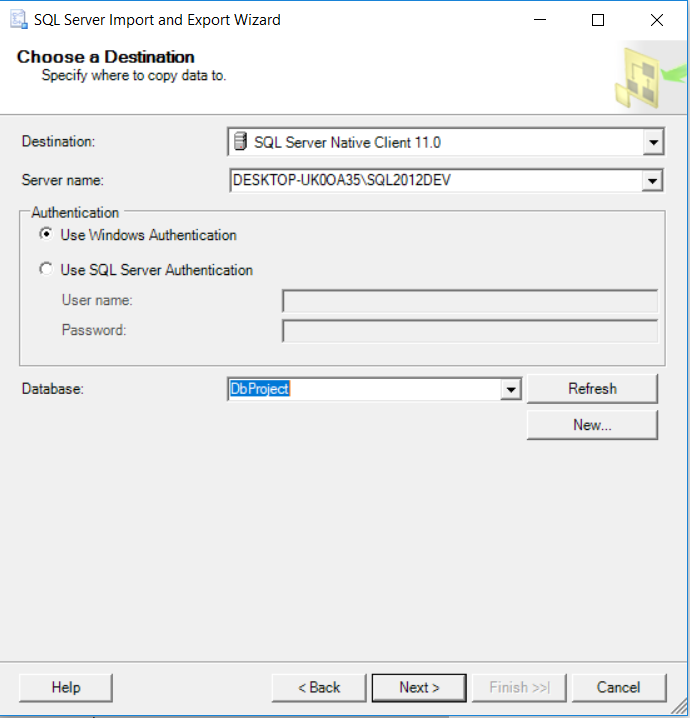
**Solution**: Default data type which the import wizard takes is string of length 50 but the PROD\_DESCRIPT column of DataSet3.txt has more than 50 characters and hence changing the data type length to 255 resolved the issue.

1. Following are the three major steps involved:

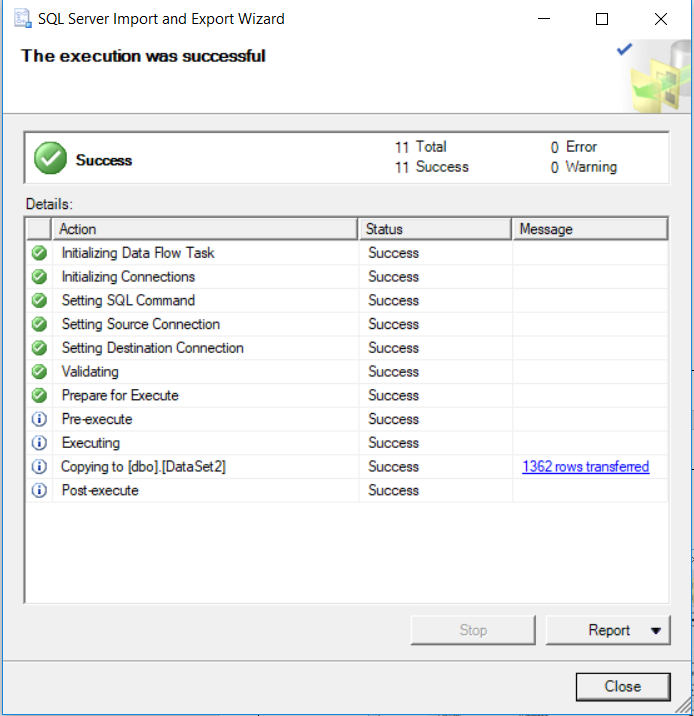
Select the data source – flat file location



Choose the destination – SQL Server name and database



Execute the data load step



By end of this step, we have all the three datasets in the staging location of our database.

1. **Data Cleansing**

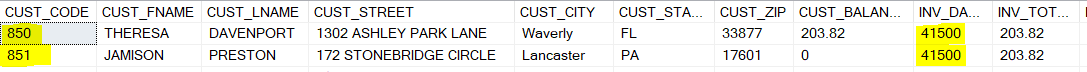
After closing analysing the data, there were several types of data errors which are to be corrected as part of data cleansing step.

Following are such categories:

1. Missing values
   1. **Issue:** VEND\_STREET and VEND\_NAME columns are empty for 6 records in **DataSet3.**

**Solution:** There are other records of the same vendor in the dataset from which these values can be filled back.

1. Wrong map of records
   1. **Issue:** It is expected that there should be a 1:1 map between a customer and an invoice but certain records in DataSet3 violate this constraint. All such invoice no’s : 1978, 2124, 2275, 2315, 2362, 2364, 2577, 2644, 2885, 2921, 3024, 3135, 3347, 3364, 3370

**Solution:** As there is no logical way to choose one of among the customers who have the same Invoice\_Num, to make the system more reliable, I choose to ignore such invoices which are mapped to multiple customers. If we aren’t sure of any data, better avoid it rather than confuse things more. We keep track of such bad-data in different table which can be rectified or verified on getting additional data points.

1. Bad characters:
   1. **Issue:** PROD\_SKU column of **DataSet3** has bad characters. Ex: 1336-FVM§

Others are: 2233-GJH?, 3393-AZQ?, 5379-BLX?, 6358-UST?, 8338-JZUû

Total records: 6

**Solution:** Delete the trailing characters which doesn’t make sense.

* 1. **Issue:** PROD\_QOH column of **DataSet3** has bad characters at the beginning of the value. Values: Æ44,?100,?88,?54,ñ51,Ä86. A total 6 records.

**Solution:** Eliminated such characters from the column

1. Flags/Null indicators:
   1. **Issue:** Sal\_End column of **DataSet4** has several values as ‘-‘, which is supposed to indicate null but having a ‘-‘ in SQL doesn’t make sense.

**Solution:** Replaced ‘-‘ with null which has a meaning in SQL

1. **Create Schema**

Once we have the data cleaned up, the next step is to define the database structures, tables, for the data at hand. The data presented in the datasets is in denormalized format, but we are expected to have the data in a production system to be in a normalized fashion to ensure data integrity and consistency. With the help of data dictionary, I created the schema for the following ten tables and its attributes:

* Brand to store brand related info
  + Brand\_Id
  + Brand\_name
  + Brand\_Type
* Product
  + Prod\_Sku
  + prod\_descript
  + prod\_type
  + prod\_base
  + prod\_Category
  + prod\_price
  + prod\_Qoh
  + prod\_min
  + brand\_Id
* Vendor
  + Vend\_Id
  + Vend\_Name
  + vend\_street
  + vend\_City
  + vend\_State
  + vend\_Zip
* Supplies
  + Vend\_Id
  + Prod\_Sku
* Customer
  + Cust\_Code
  + Cust\_Fname
  + Cust\_Lname
  + Cust\_Street
  + Cust\_State
  + Cust\_City
  + Cust\_Zip
  + Cust\_Balance
* Department
  + Dept\_Num
  + Dept\_Name
  + Dept\_Mail\_Box
  + Dept\_Phone
  + Emp\_num
* Employee
  + Emp\_num
  + Emp\_Fname
  + Emp\_Lname
  + Emp\_Email
  + Emp\_Phone
  + Emp\_HiredDate
  + Emp\_Title
  + Emp\_Comm
  + Dept\_Num
* Salary\_History
  + Emp\_num
  + Sal\_from
  + Sal\_end
  + sal\_Amount
* Invoice
  + Inv\_Num
  + Cust\_Code
  + Inv\_Date
  + Inv\_total
  + Employee\_Id
* Line
  + Inv\_Num
  + Line\_Num
  + Line\_Qty
  + Line\_price
  + Prod\_Sku

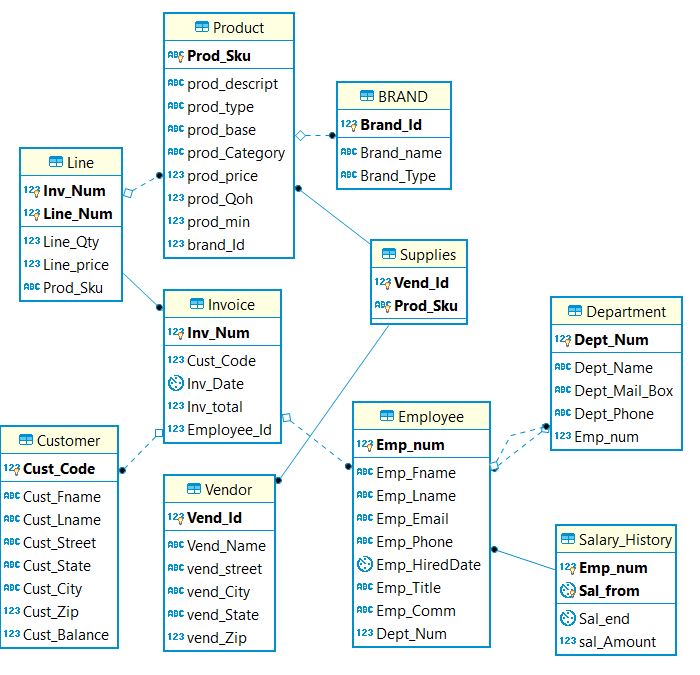
1. **Define primary keys and foreign keys**

The next logical step is to define primary keys and foreign keys to maintain data and referential integrity. Following is the list of such constraints at table-level:

* Brand
  + PRIMARY KEY: Brand\_Id
* Customer
  + PRIMARY KEY: Cust\_Code
* Department
  + PRIMARY KEY: Dept\_Num
  + FOREIGN KEY: Emp\_Num REFERENCES Employee(Emp\_num)
* Employee
  + PRIMARY KEY: Emp\_Num
  + FOREIGN KEY: Dept\_Num REFERENCES Department(Dept\_num)
* Invoice
  + PRIMARY KEY: Inv\_Num
  + FOREIGN KEY: Cust\_code REFERENCES Customer(Cust\_Code)
  + FOREIGN KEY: Employee\_Id REFERENCES Employee(Emp\_Num)
* Line
  + PRIMARY KEY: Inv\_Num, Line\_Num
  + FOREIGN KEY: Prod\_Sku REFERENCES Product(Prod\_Sku)
  + FOREIGN KEY: Inv\_Num REFERENCES Invoice(Inv\_Num)
* Product
  + PRIMARY KEY: Prod\_SKU
  + FOREIGN KEY: Brand\_Id REFERENCES Brand(Brand\_Id)
* Salary\_History
  + PRIMARY KEY: Emp\_num, Sal\_from
  + FOREIGN KEY: Emp\_Num REFERENCES Employee(Emp\_Num)
* Supplies
  + PRIMARY KEY: Vend\_Id, Prod\_Sku
  + FOREIGN KEY: Vend\_Id REFERENCES Vendor(Vend\_Id)
  + FOREIGN KEY: Prod\_Sku REFERENCES Product(Prod\_Sku)
* Vendor
  + PRIMARY KEY: Vendor\_Id

1. **ER model**

Using DBeaver, I have connected to the MS SQL Server local instance and generated a ER model for the database.



1. **Data load into Actual tables**

Once the tables and constraints are created, I have loaded the data from the transformed datasets into individual tables by using the code attached below.

1. **Write queries**

1. Write a query to display the current salary for each employee in department 300. Assume that only current employees are kept in the system, and therefore the most current salary for each employee is the entry in the salary history with a NULL end date. Sort the output in descending order by salary amount.

**Query:**

SELECT E.\*,

S.sal\_amount

FROM employee E

INNER JOIN salary\_history S

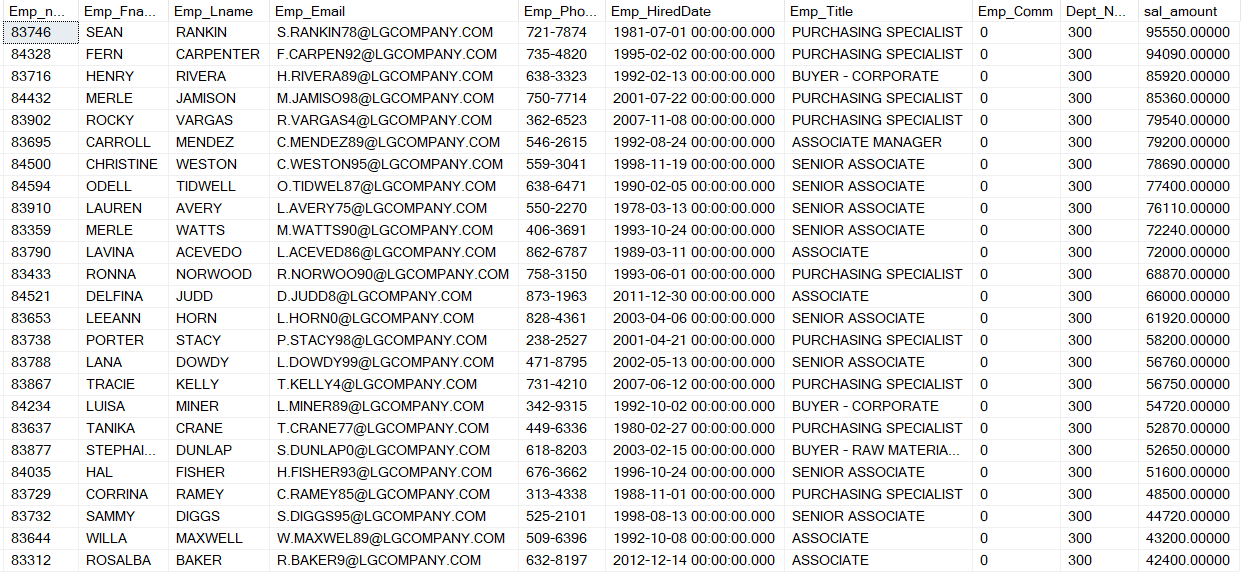
ON E.emp\_num = S.emp\_num

WHERE E.dept\_num = 300

AND S.sal\_end IS NULL

ORDER BY S.sal\_amount DESC;

**Output: (total of 25 records)**



2. Write a query to display the starting salary for each employee. The starting salary would be the entry in the salary history with the oldest salary start date for each employee. Sort the output by employee number.

**Query:**

SELECT E.\*,

S.sal\_amount

FROM employee E

INNER JOIN salary\_history S

ON E.emp\_num = S.emp\_num

INNER JOIN (SELECT emp\_num,

Min(sal\_from) Min\_Sal\_From

FROM salary\_history

GROUP BY emp\_num) AS SM

ON E.emp\_num = SM.emp\_num

WHERE S.sal\_from = SM.min\_sal\_from

ORDER BY E.emp\_num;

**Output: (total of 363 records)**



3. Write a query to display the invoice number, line numbers, product SKUs, product descriptions, and brand ID for sales of sealer and top coat products of the same brand on the same invoice.

**Query:**

SELECT L1.inv\_num,

L1.line\_num,

L2.line\_num,

P1.prod\_sku,

P2.prod\_sku,

P1.prod\_descript,

P2.prod\_descript,

P1.brand\_id

FROM line L1

INNER JOIN product P1

ON L1.prod\_sku = P1.prod\_sku

INNER JOIN line L2

ON L1.inv\_num = L2.inv\_num

INNER JOIN product P2

ON L2.prod\_sku = P2.prod\_sku

AND P1.brand\_id = P2.brand\_id

WHERE P1.prod\_category = 'Sealer'

AND P2.prod\_category = 'Top Coat'

ORDER BY L1.inv\_num,

L2.line\_num;

**Output: (0 records)**

4. The Binder Prime Company wants to recognize the employee who sold the most of their products during a specified period. Write a query to display the employee number, employee first name, employee last name, e-mail address, and total units sold for the employee who sold the most Binder Prime brand products between November 1, 2015, and December 5, 2015. If there is a tie for most units sold, sort the output by employee last name.

**Query:**

SELECT E.emp\_num,

E.emp\_fname,

E.emp\_lname,

E.emp\_email,

M.unitscount

FROM employee AS E

INNER JOIN (SELECT I.employee\_id,

Sum(line\_qty) AS UnitsCount

FROM invoice i

INNER JOIN line l

ON I.inv\_num = l.inv\_num

INNER JOIN product p

ON l.prod\_sku = p.prod\_sku

INNER JOIN brand B

ON b.brand\_id = p.brand\_id

WHERE brand\_name = 'BINDER PRIME'

AND I.inv\_date BETWEEN '2015-11-01' AND '2015-12-05'

GROUP BY I.employee\_id) M

ON E.emp\_num = M.employee\_id

WHERE unitscount = (SELECT Max(unitscount)

FROM (SELECT I.employee\_id,

Sum(line\_qty) AS UnitsCount

FROM invoice I

INNER JOIN line L

ON I.inv\_num = L.inv\_num

INNER JOIN product P

ON l.prod\_sku = p.prod\_sku

INNER JOIN brand B

ON B.brand\_id = P.brand\_id

WHERE B.brand\_name = 'BINDER PRIME'

AND inv\_date BETWEEN '2015-11-01' AND

'2015-12-05'

GROUP BY employee\_id) AS E)

ORDER BY E.emp\_lname;

**Output: (0 records)**

5. Write a query to display the customer code, first name, and last name of all customers who have had at least one invoice completed by employee 83649 and at least one invoice completed by employee 83677. Sort the output by customer last name and then first name.

**Query:**

SELECT C.cust\_code,

C.cust\_fname,

C.cust\_lname

FROM customer C

INNER JOIN invoice I

ON C.cust\_code = I.cust\_code

WHERE I.employee\_id IN ( 83649, 83677 )

GROUP BY C.cust\_code,

C.cust\_fname,

C.cust\_lname

HAVING Count(DISTINCT I.employee\_id) = 2

ORDER BY 1;

**Output: (0 records)**

6. LargeCo is planning a new promotion in Alabama (AL) and wants to know about the largest purchases made by customers in that state. Write a query to display the customer code, customer first name, last name, full address, invoice date, and invoice total of the largest purchase made by each customer in Alabama. Be certain to include any customers in Alabama who have never made a purchase (their invoice dates should be NULL and the invoice totals should display as 0).

**Query:**

SELECT C.cust\_code,

C.cust\_fname,

C.cust\_lname,

C.cust\_street,

C.cust\_city,

C.cust\_state,

C.cust\_zip,

I.inv\_date,

Isnull(I.inv\_total, 0) AS LargestPurchase

FROM customer C

INNER JOIN invoice I

ON C.cust\_code = I.cust\_code

WHERE C.cust\_state = 'AL'

AND I.inv\_total = (SELECT Max(IM.inv\_total)

FROM invoice IM

WHERE IM.cust\_code = C.cust\_code)

UNION ALL

SELECT C.cust\_code,

C.cust\_fname,

C.cust\_lname,

C.cust\_street,

C.cust\_city,

C.cust\_state,

C.cust\_zip,

NULL,

0

FROM customer c

WHERE C.cust\_state = 'AL'

AND C.cust\_code NOT IN (SELECT cust\_code

FROM invoice)

ORDER BY 1;

**Output: (Total of 50 records)**



7. One of the purchasing managers is interested in the impact of product prices on the sale of products of each brand. Write a query to display the brand name, brand type, average price of products of each brand, and total units sold of products of each brand. Even if a product has been sold more than once, its price should only be included once in the calculation of the average price. However, you must be careful because multiple products of the same brand can have the same price, and each of those products must be included in the calculation of the brand’s average price.

**Query:**

SELECT brand\_name,

brand\_type,

PR.avg\_price AS AveragePrice,

UN.units\_sold AS UnitsSold

FROM brand b

INNER JOIN (SELECT brand\_id,

Avg(prod\_price) AS avg\_price

FROM product

GROUP BY brand\_id) PR

ON b.brand\_id = PR.brand\_id

INNER JOIN (SELECT brand\_id,

Sum(line\_qty) AS units\_sold

FROM product p

INNER JOIN line l

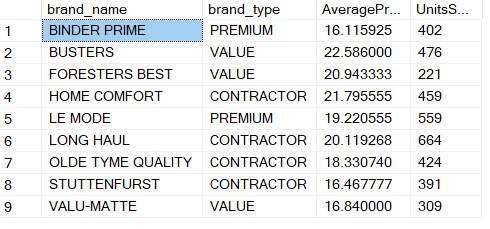
ON p.prod\_sku = l.prod\_sku

GROUP BY brand\_id) UN

ON b.brand\_id = UN.brand\_id

ORDER BY brand\_name;

**Output: (Total of 9 records)**



8. The purchasing manager is still concerned about the impact of price on sales. Write a query to display the brand name, brand type, product SKU, product description, and price of any products that are not a premium brand, but that cost more than the most expensive premium brand products.

**Query:**

SELECT b.brand\_name,

b.brand\_type,

p.prod\_sku,

p.prod\_descript,

p.prod\_price

FROM product p

INNER JOIN brand b

ON p.brand\_id = b.brand\_id

WHERE brand\_type != 'PREMIUM'

AND prod\_price > (SELECT Max(prod\_price)

FROM product P

INNER JOIN brand B

ON P.brand\_id = B.brand\_id

WHERE B.brand\_type = 'PREMIUM')

ORDER BY 1;

**Output: (Total of 1 record)**



9. Using SQL descriptive statistics functions calculate the value of the following items:

a. What are the products that have a price greater than $50?

**Query:**

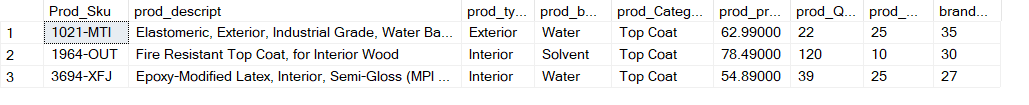
SELECT \*

FROM product P

WHERE P.prod\_price > 50

ORDER BY 1;

**Output: (Total of 3 records)**



b. What is total value of our entire inventory on hand?

**Query:**

SELECT Sum(prod\_qoh \* prod\_price) As TotalCost

FROM product P;

**Output: (1 record)**



c. How many customers do we presently have and what is the total of all customer balances?

**Query:**

SELECT Count(\*) AS CustomerCount,

Sum(cust\_balance) AS TotalBalance

FROM customer;

**Output: (1 record)**



d. What are to top three states that buy the most product in dollars from the company?

**Query:**

SELECT TOP 3 C.cust\_state,

Sum(I.inv\_total) AS TotalPurchase

FROM customer C

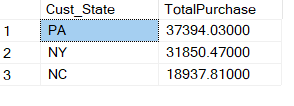
INNER JOIN invoice I

ON I.cust\_code = C.cust\_code

GROUP BY C.cust\_state

ORDER BY 2 DESC;

**Output: (Total of 3 records)**



1. **Connect to SQL Server from R**

I used **RODBC** library to connect to MS SQL Server from R. Snipped of the code is as follows:

library(RODBC)

dbconnection <- odbcDriverConnect("connectionstring")

data <- sqlQuery(dbconnection, paste("select Inv\_date AS date, SUM(Inv\_total) As total from Invoice group by inv\_date;"))

odbcClose(dbconnection)

1. **Define and run regression models**

Once, we import the data into R, we can define a regression model to predict inventory total.

1. **Forecast data**

10. Using predictive statistics calculate what the predicted forecast of sales for the next year based on the INV\_DATE (independent) and INV\_TOTAL (dependent). Remember that you will need to convert the INV\_DATE from the MS SQL Server stored date value to the expect Julian date, since numbers in MS SQL are stored as the number of days since 1/1/1900 with the fraction as the portion of a day (if you are using a different DBMS use the appropriate code for conversion.)