BUAN 6320 Project – Dilip Merala

I used Microsoft SQL Server for this project.

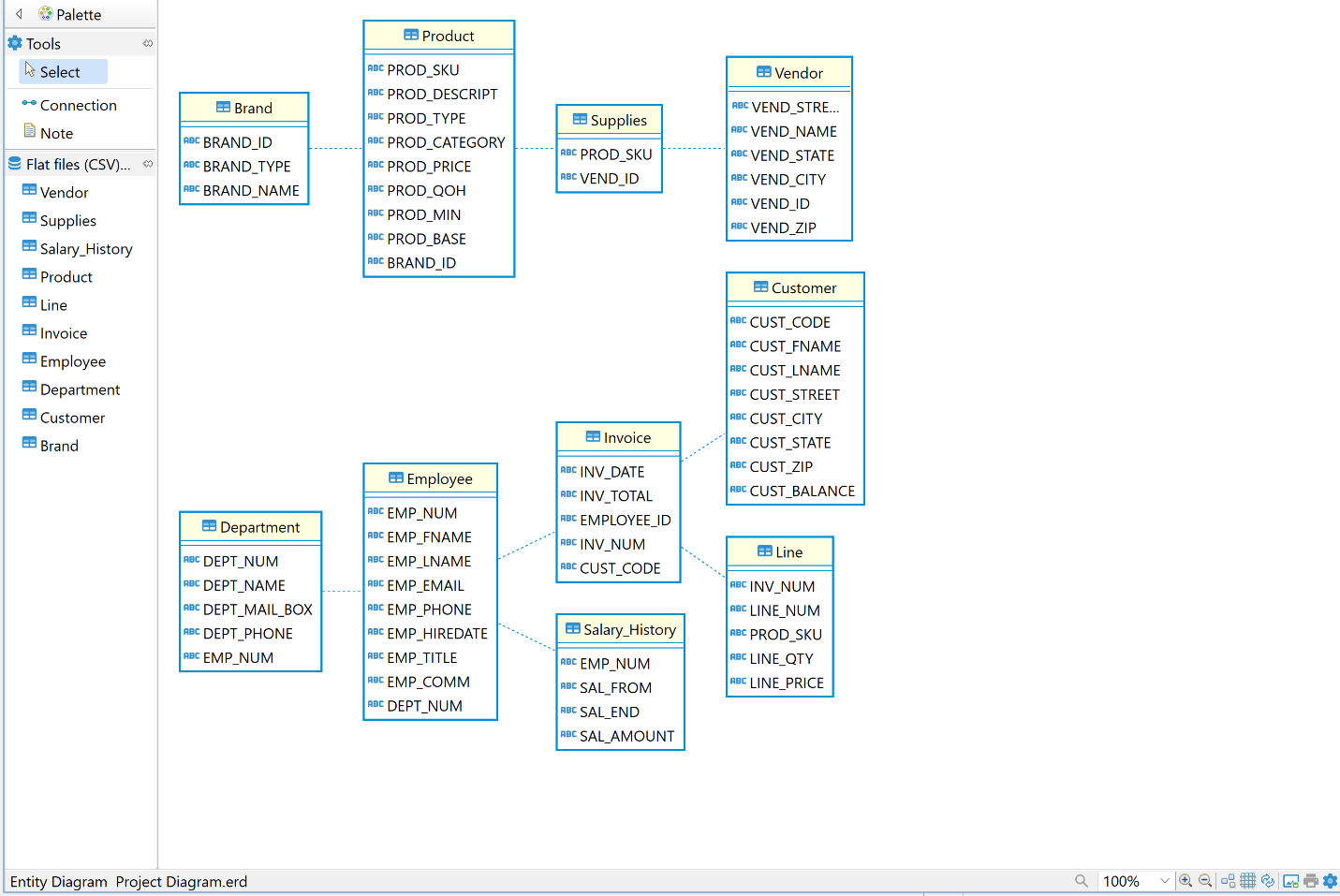
**Ans 1: Normalization**

I imported all the data into Excel and created tales based on Normalization in order to meet the Data Schema requirements.

**Ans 2: Data Munging**

I replaced the blank spaces with nulls. I created 9 tables in the csv format and one (Product) in the Tab delimited format as I faced errors while importing it. I realized that this was because of the “Product\_Descript” column which cannot be used as comma separated. So, I saved the file in tab delimited format and then imported it in my MSSQL server. I also changed the limit to 150 characters as I was getting a truncation error while importing it in the tab delimited format.

**Ans 3: ERD created in DBeaver**



**Ans 4: Keys**

**Brand:**

Primary –> Brand\_ID

**Product:**

Primary -> PROD\_SKU  
Secondary -> Brand\_ID

**Supplies**

Primary -> PROD\_SKU  
Secondary -> VEND\_ID

**Vendor:**

Primary -> VEND\_ID

**Department:**

Primary -> DEPT\_NUM  
Secondary -> EMP\_NUM

**Employee:**

Primary -> EMP\_NUM  
Secondary -> DEPT\_NUM

**Invoice:**

Primary -> INV\_NUM  
Secondary -> CUST\_CODE

**Salary\_History:**

Secondary -> EMP\_NUM

**Customer:**

Primary -> CUST\_CODE

**Line:**

Primary -> LINE\_NUM  
Secondary -> INV\_NUM

**Ans 5:**

**Here are the Queries and output:**

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.

SQL Query:

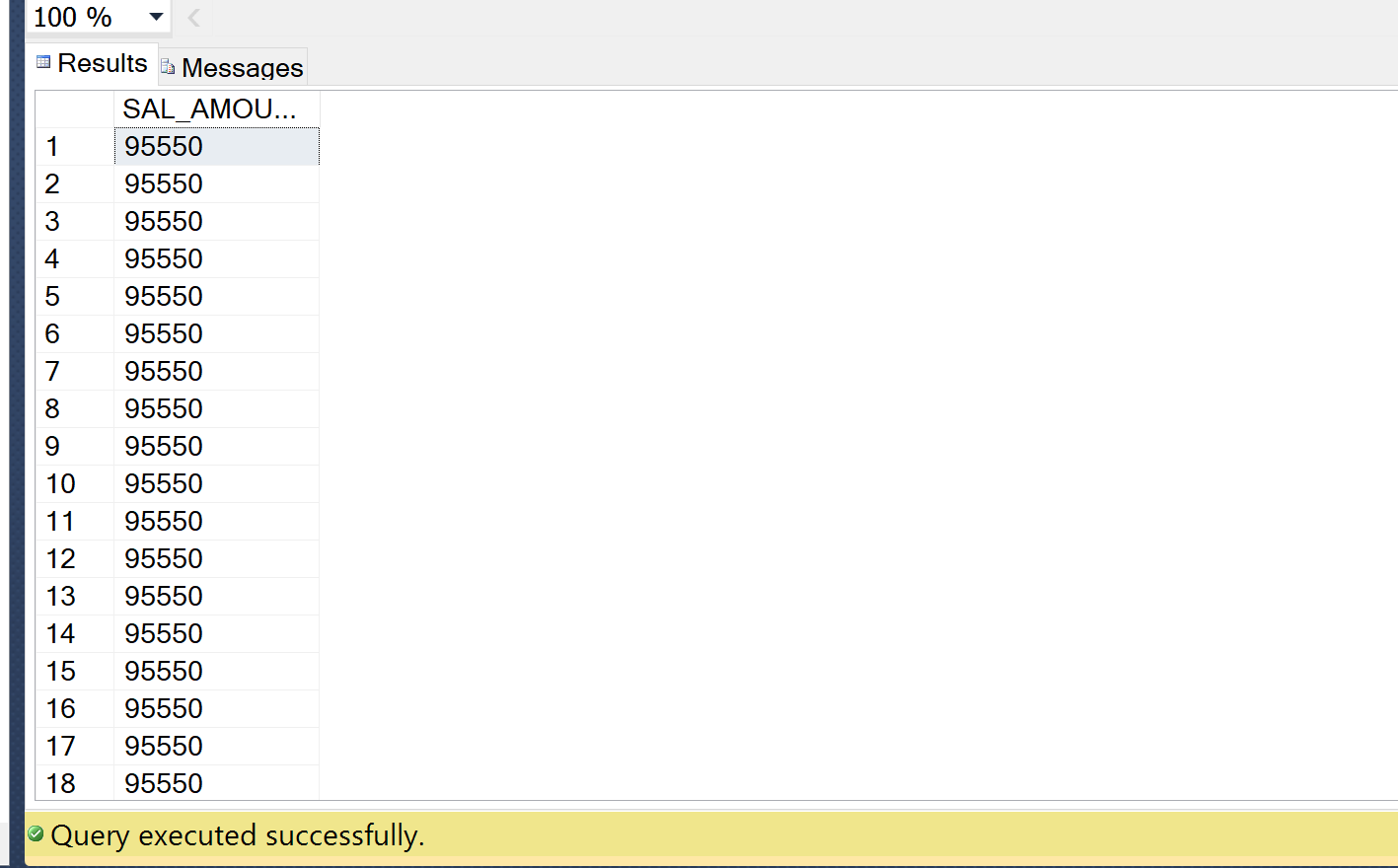
use Project

select h.SAL\_AMOUNT, h.EMP\_NUM From Salary\_History h JOIN Department d ON h.EMP\_NUM = d.EMP\_NUM

WHERE d.DEPT\_NUM = 300 AND h.SAL\_END = ''

ORDER BY h.SAL\_AMOUNT DESC

Result:



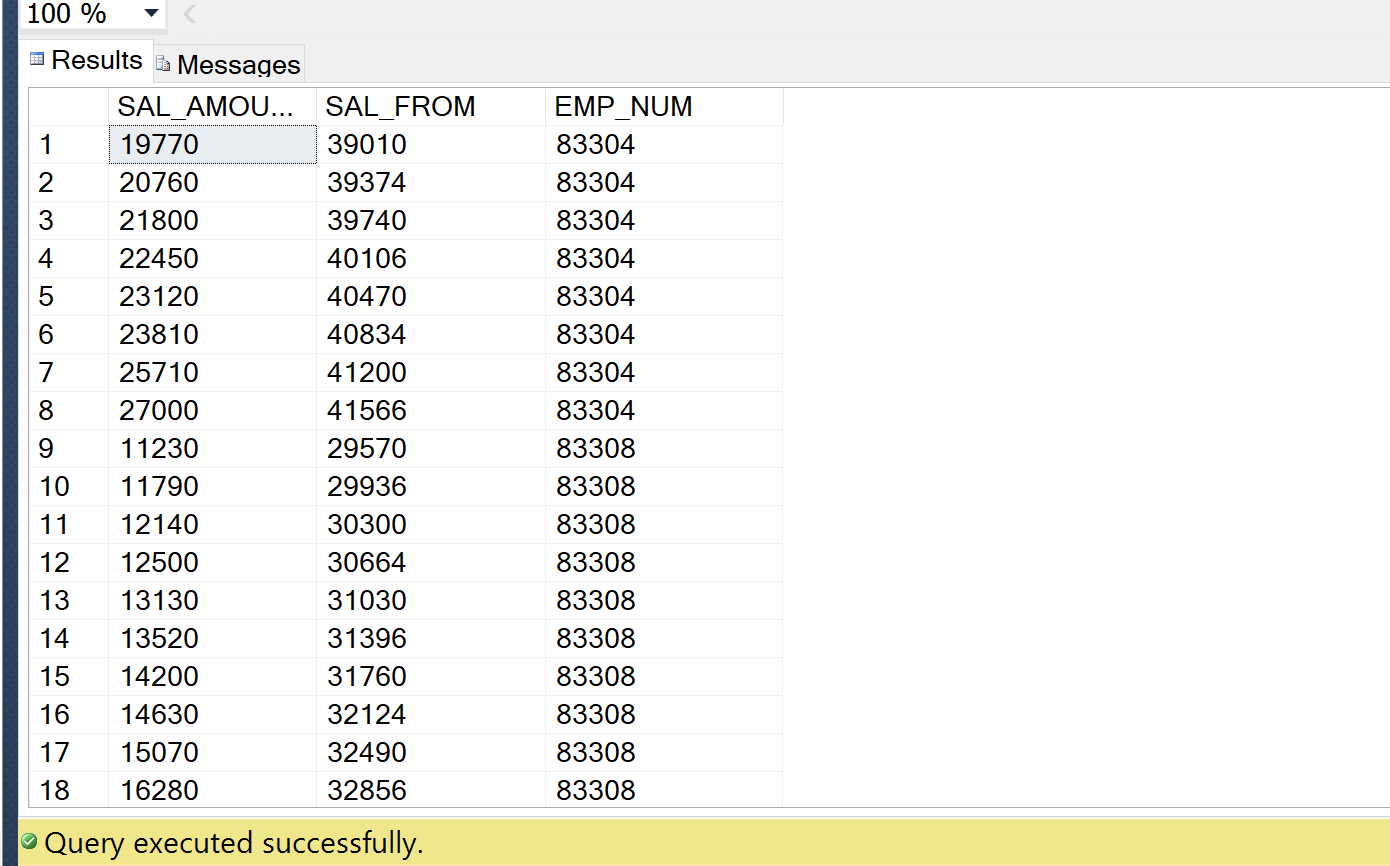
1. Write a query to display the starting salary for each employee. The starting salary would be the entry in the salary history with the oldest salary start date for each employee. Sort the output by employee number.

SQL Query:

SELECT h.SAL\_AMOUNT, h.SAL\_FROM, h.EMP\_NUM from Salary\_History h

ORDER BY h.EMP\_NUM, h.SAL\_FROM ASC;

Output:



1. 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.

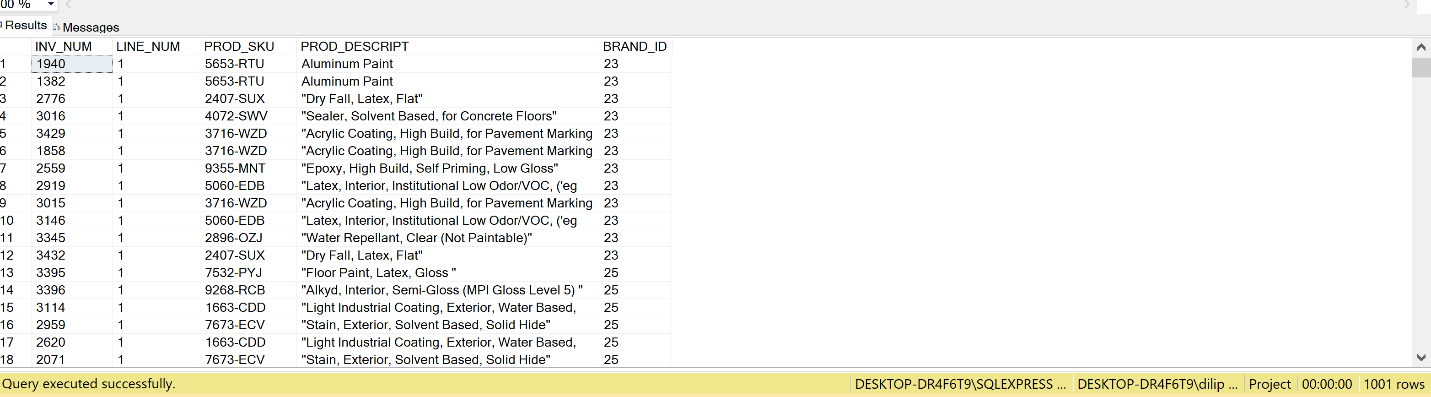
SQL Query:

SELECT l.INV\_NUM, l.LINE\_NUM, p.PROD\_SKU, p.PROD\_DESCRIPT, p.BRAND\_ID FROM Product p JOIN Line l ON l.PROD\_SKU = p.PROD\_SKU

WHERE p.PROD\_CATEGORY LIKE 'Sealer' OR p.PROD\_CATEGORY LIKE 'Top Coat'

ORDER BY l.LINE\_NUM ASC, p.BRAND\_ID

Output:



1. 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.

SQL Query:

SELECT DISTINCT e.EMP\_NUM, e.EMP\_FNAME, e.EMP\_LNAME, e.EMP\_EMAIL, l.LINE\_QTY FROM Employee e JOIN Invoice i ON e.EMP\_NUM = i.EMPLOYEE\_ID

JOIN LINE l ON i.INV\_NUM = l.INV\_NUM

JOIN PRODUCT p ON l.PROD\_SKU = p.PROD\_SKU

JOIN Brand b ON p.BRAND\_ID = b.BRAND\_ID

WHERE b.BRAND\_ID = '33' AND i.INV\_DATE BETWEEN '2015-11-01' AND '2015-12-05'

ORDER BY l.LINE\_QTY DESC, e.EMP\_LNAME

Output: No results found.

1. 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.

SQL Query:

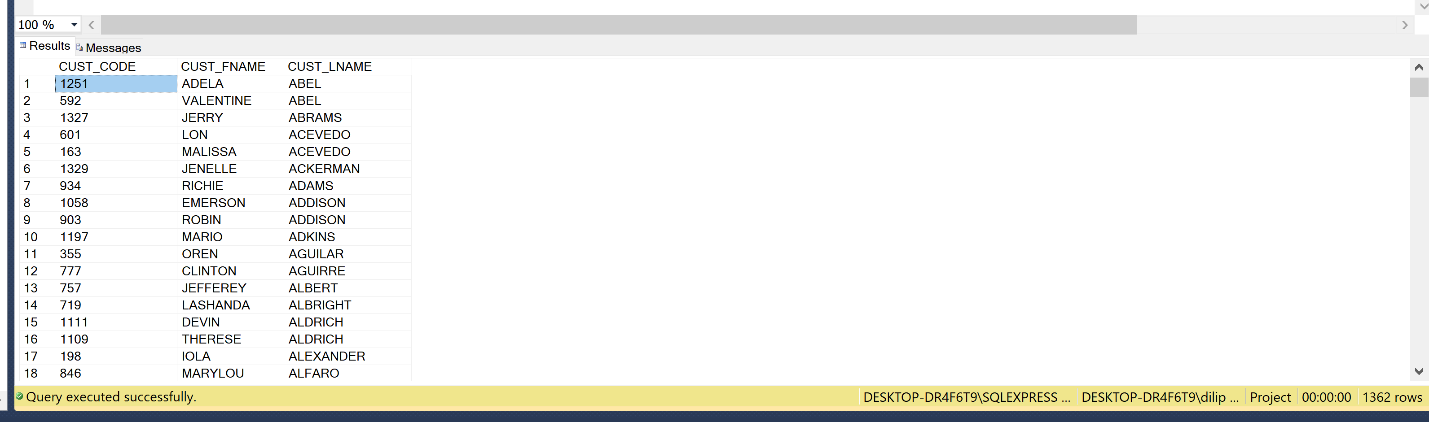
SELECT DISTINCT c.CUST\_CODE, c.CUST\_FNAME, c.CUST\_LNAME FROM EMPLOYEE e JOIN Invoice i ON e.EMP\_NUM = i.EMPLOYEE\_ID

JOIN Customer c ON i.CUST\_CODE = c.CUST\_CODE

WHERE EXISTS (SELECT i.EMPLOYEE\_ID FROM Invoice i WHERE i.EMPLOYEE\_ID = 83649 OR i.EMPLOYEE\_ID = 83677)

ORDER BY c.CUST\_LNAME, c.CUST\_FNAME

Output:



1. 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).

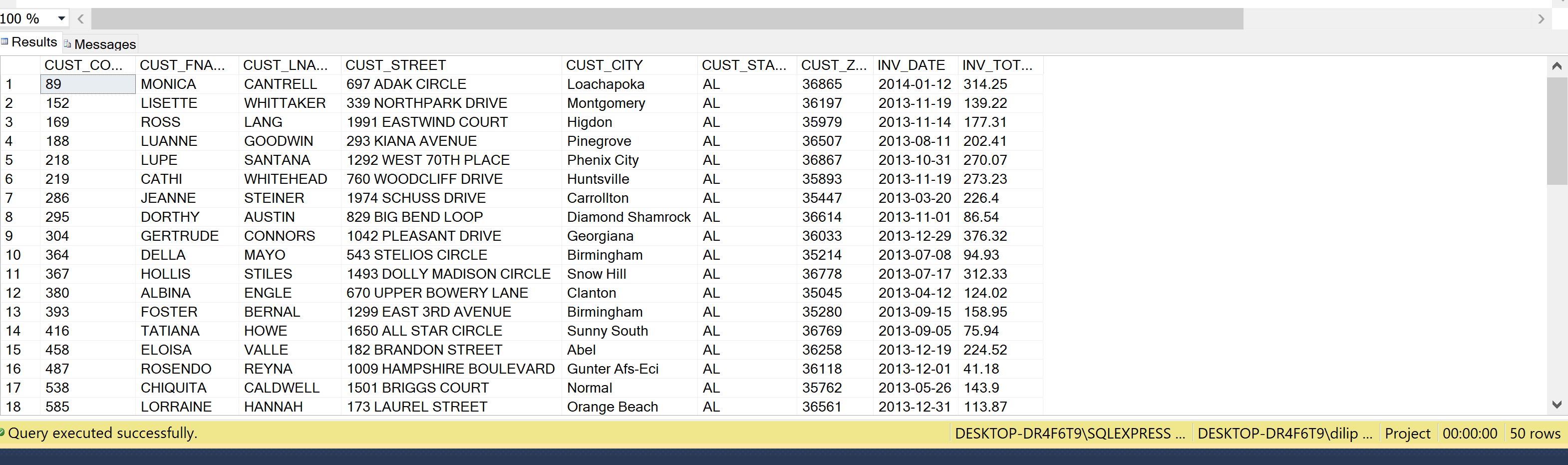
SQL Query:

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, i.INV\_TOTAL

FROM Customer c JOIN INVOICE i ON c.CUST\_CODE = i.CUST\_CODE

WHERE CUST\_STATE = 'AL'

Output:



1. 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.

SQL Query:

ALTER TABLE PRODUCT

ALTER COLUMN Prod\_Price decimal

ALTER TABLE LINE

ALTER COLUMN LINE\_QTY decimal

SELECT b.BRAND\_NAME, b.BRAND\_TYPE, AVG(p.PROD\_PRICE) AS AVGP, new\_tab.Total\_Sold

FROM PRODUCT p LEFT JOIN Brand b ON p.BRAND\_ID = b.BRAND\_ID

INNER JOIN (select p.BRAND\_ID, SUM(l.LINE\_QTY) as 'Total\_Sold'

FROM line l LEFT OUTER JOIN product p ON l.PROD\_SKU = p.PROD\_SKU

GROUP BY p.BRAND\_ID) as new\_tab

ON b.BRAND\_ID = new\_tab.BRAND\_ID

group by p.BRAND\_ID

1. 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.

SQL Query:

SELECT b.BRAND\_NAME, b.BRAND\_TYPE,p.Prod\_SKU, PROD\_DESCRIPT, PROD\_PRICE

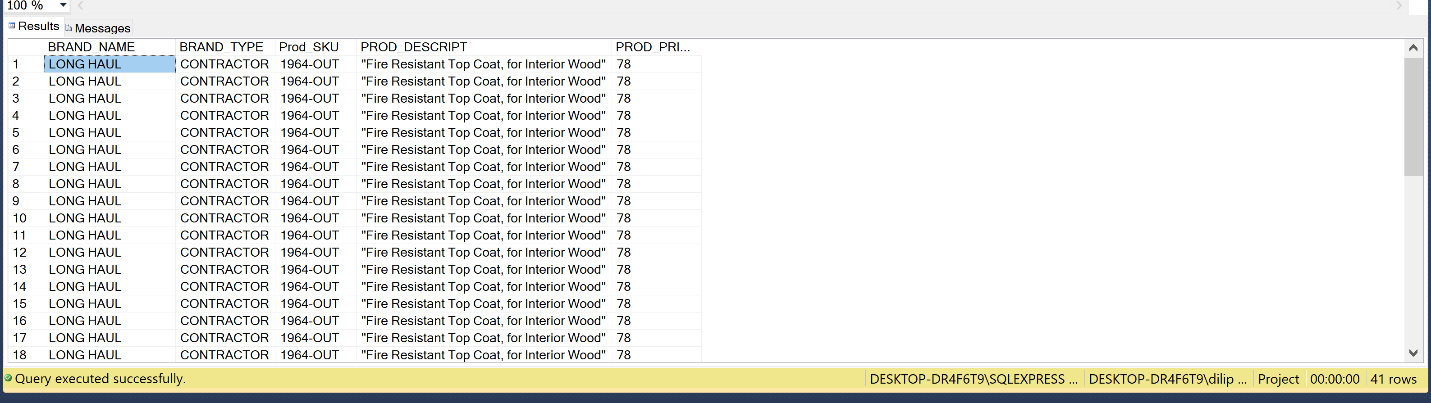
FROM Product p JOIN Brand b ON p.BRAND\_ID = b.BRAND\_ID

WHERE b.BRAND\_TYPE NOT LIKE 'PREMIUM' AND p.PROD\_PRICE > (SELECT MAX(p.PROD\_PRICE)

FROM product p JOIN brand b ON p.BRAND\_ID = b.BRAND\_ID

WHERE BRAND\_TYPE = 'PREMIUM')

Output:



1. Using SQL descriptive statistics functions calculate the value of the following items:
   1. What are the products that have a price greater than $50?

select \* from Product p

WHERE p.PROD\_PRICE > 50

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

select SUM(PROD\_PRICE) from Product p  
  
ANS: 4873

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

select COUNT(CUST\_CODE) FROM Customer c

ANS: 1362

ALTER TABLE CUSTOMER

ALTER Column CUST\_BALANCE decimal

select SUM(CUST\_BALANCE) FROM Customer

ANS: 787211

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

select CUST\_STATE, LINE\_QTY from

Customer JOIN Invoice ON Customer.CUST\_CODE = Invoice.CUST\_CODE

JOIN LINE ON Invoice.INV\_NUM = Line.INV\_NUM

JOIN Product on Product.PROD\_SKU = Line.PROD\_SKU

ORDER BY LINE\_QTY DESC;

ANS: NC, MI and FL

1. 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.)

declare @d1 datetime

set @d1 = 41867

select @d1

select CONVERT(varchar(20),@d1,120)

or if you want to do it in one statement:

select CONVERT(varchar(25),cast(41867 as datetime),120)

Analyze your results from the linear regression, and provide the R2, model, coefficients, and the confidence interval for your analysis.

Solution: I got **R-squared value of 0.3634** when I ran the following code in R Management studio:

projDB.df <- read.csv("Invoice.csv")

library(tidyverse)

library(forecast)

library(leaps)

library(caret)

#Partitioning Data

set.seed(1) # set seed for reproducing the partition

train.index <- sample(c(1:1000), 600)

train.df <- projDB.df[train.index, ]

valid.df <- projDB.df[-train.index, ]

#Run regression

proj.lm <- lm(INV\_TOTAL ~ ., data = train.df)

options(scipen = 999)

summary(proj.lm)

Since the **R-squared value of 0.3634** is quite low, the data needs to be better organized and more samples need to be obtained to work with other variables.