ANH LE

BUAN 6320.002

Professor James Scott

Dec 12, 2018

**INDIVIDUAL PROJECT**

1. **PROJECT BACKGROUND**

In this project, by using three given datasets and the Data Dictionary, I split the data into ten separate tables which are named BRAND, CUSTOMER, DEPARTMENT, EMPLOYEE, INVOICE, LINE, PRODUCT, SALARY\_HISTORY, SUPPLIES and VENDOR. My next step is to clean the provided data.

1. **DATA CLEANING**

After separating the datasets into ten different tables, I checked the data in each table. The following are the problems I found in each table and how I solved these problems. I executed the data cleaning process by using Microsoft Excel.

1. In the BRAND table, there are 3 fields which are BRAND\_ID, BRAND\_TYPE, and BRAND\_NAME. There were no missing values. I grouped the data presented in this table by the BRAND\_ID.
2. In the CUSTOMER table, there are 8 fields which are CUST\_CODE, CUST\_FNAME, CUST\_LNAME, CUST\_STREET, CUST\_CITY, CUST\_STATE, CUST\_ZIP and CUST\_BALANCE. There were no missing values or values with bad characters in this table. The primary key in this table is CUST\_CODE.
3. The DEPARTMENT table has 5 fields which are DEPT\_NUM, DEP\_NAME, DEPT\_MAILBOX, DEPT\_PHONE, and EMPT\_NUM. I grouped the data by DEPT\_NUM, which is the primary key of this table.
4. In the EMPLOYEE table, there are fields named EMP\_NUM, EMP\_FNAME, EMP\_LNAME, EMP\_EMAIL, EMP\_PHONE, EMP\_HIREDATE, EMP\_TITLE, EMP\_COMM, DEPT\_NUM, and SUPV\_EMP\_NUM. I grouped the data by EMP\_NUM, which is the primary key of this table. I kept SUPV\_EMP\_NUM because this attribute show which employees are supervisors in each department.
5. In the INVOICE table, there are 5 fields which are INV\_NUM, INV\_DATE, INV\_TOTAL, CUST\_CODE, and EMPLOYEE\_ID. The EMPLOYEE\_ID in this table and the EMP\_NUM refer to the same employee. In the INVOICE table, I found a major problem. By rule, each invoice is generated by only one customer. However, in the given data, there are some invoices that have more than one customer.

* Invoice number 1978, 2124, 2275, 2315, 2362, 2364, 2577, 2644, 2885, 2921, 3024, 3135, 3347, 3364, 3370 had two customer codes. The following table are details of those records and my observation of the duplicate data.

|  |  |  |
| --- | --- | --- |
| **INV\_NUM** | **CUST\_CODE** | **My observation** |
| 1978 | 850, 851 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2124 | 917, 918 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2275 | 392, 393 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2315 | 993, 994 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2362 | 1014, 1015 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2364 | 877, 878 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2577 | 729, 730 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2644 | 988, 989 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2885 | 1218, 1219 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 2921 | 888, 889 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 3024 | 1207, 1208 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 3135 | 1307, 1308 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 3347 | 1400, 1401 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 3364 | 1410, 1411 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |
| 3370 | 928, 929 | These CUST\_CODE have the same INV\_DATE, INV\_TOTAL, EMP\_ID |

* Even though these invoices had two customer codes, the information related to INV\_DATE, INV\_TOTAL, EMP\_ID are the same for both two customer codes. Therefore, I decided to delete one redundant data entry for each invoice which had two customers. So, after this process, all INV\_NUM had only one CUST\_CODE.
* In the INVOICE table, the INV\_DATE is in Julian format. I decided to keep and work with this date format in my project because this is a more convenient format for me to work with this.

1. In the LINE table, there are fields named INV\_NUM, LINE\_NUM, PROD\_SKU, LINE\_QTY, and LINE\_PRICE. This table has the composite primary key of INV\_NUM and LINE\_NUM. Again, in this table, I discovered some duplicate data.

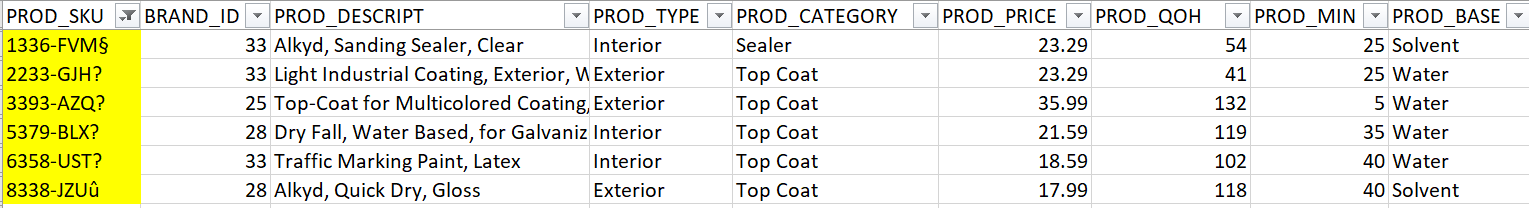
* The following table shows the invoice line which has duplicate data. My next step was to delete the redundant entity instance to make sure that I only had unique combinations of INV\_NUM and LINE\_NUM.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INV\_NUM | LINE\_NUM | PROD\_SKU | LINE\_QTY | LINE\_PRICE |
| 2275 | 2 | 5413-TTF | 2 | 29.99 |
| 2577 | 3 | 3488-GSE | 1 | 10.99 |
| 1978 | 5 | 3061-DOI | 5 | 5.59 |
| 2364 | 5 | 7673-ECV | 5 | 11.99 |
| 2921 | 1 | 1336-FVM | 3 | 23.29 |
| 2124 | 3 | 6451-KGJ | 2 | 6.99 |
| 3370 | 2 | 8094-EYX | 5 | 14.59 |
| 2644 | 2 | 9138-QCV | 2 | 21.59 |
| 2315 | 3 | 1871-GWZ | 4 | 24.99 |
| 2362 | 2 | 5587-MNY | 3 | 14.69 |
| 3024 | 2 | 2496-QMW | 3 | 38.99 |
| 2885 | 1 | 6041-PBS | 1 | 7.49 |
| 3135 | 2 | 5465-YYG | 2 | 8.99 |
| 3347 | 4 | 9288-IRF | 4 | 14.59 |
| 3364 | 3 | 3161-XJS | 1 | 38.99 |

1. In the PRODUCT table, we have PROD\_SKU, BRAND\_ID, PROD\_DESCRIPT, PROD\_TYPE, PROD\_CATEGORY, PROD\_PRICE, PROD\_QOH, PROD\_MIN, and PROD\_BASE. After checking the data in this table, here were the problems that I recognized:

* There were some funny characters under the PROD\_QOH column. To clean the data, I decided to get rid of these funny characters. Here are the results of my changes.

|  |  |
| --- | --- |
| **PROD\_QOH before** | **PROD\_QOH after** |
| Æ44 | 44 |
| ?100 | 100 |
| ?88 | 88 |
| ?54 | 54 |
| ñ51 | 51 |
| Ä86 | 86 |

* I also found some funny characters in the PROD\_SKU column. Hence, again I decided to get rid of these funny characters.

The figure below presented the PROD\_SKUs after the changes.

1. In the SALARY\_HISTORY table, the fields are EMP\_NUM, SAL\_FROM, SAL\_END, and SAL\_AMOUNT. The primary key in this table is comprised of EMP\_NUM and SAL\_FROM.

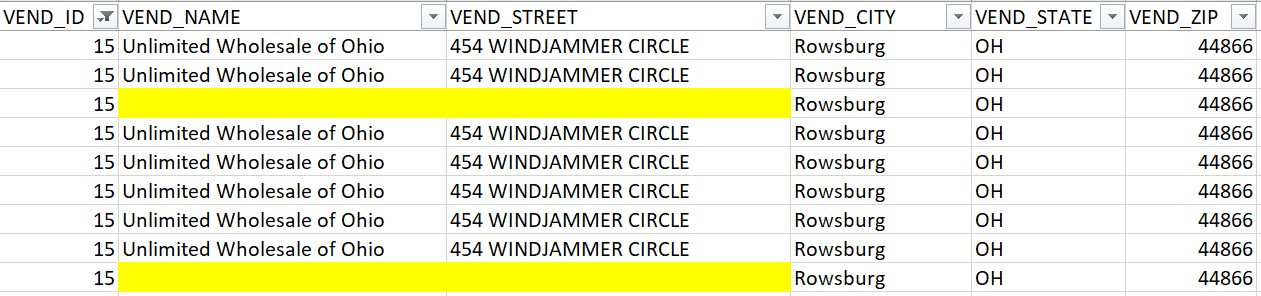
* The SAL\_FROM and SAL\_END are dates in Julian date format. I decided to keep the data as is because I think such format was easier to work with than the normal date format (YYYY-MM-DD or MM/DD/YYY, etc.).
* In the SAL\_END attribute, there are some “-“ (dashes) which indicate that the employee is currently still receiving the salary amount corresponding to the specific SAL\_FROM date. However, as we have different types for dashes: hyphen (-), en dash (–), and em dash (—), using “-“ as a flag can cause some data input errors. Thus, I decided to change these dashes to “-99” because I think it is a better flag for the database.

1. In the SUPPLIES table, there are 2 fields which are PROD\_SKU and VEND\_ID. The primary key of this table is comprised of both PROD\_SKU and VEND\_ID.

* In this table, under the PROD\_SKU attribute, I found the same funny characters as I did in the PRODUCT table. Therefore, I deleted these characters from the PROD\_SKUs.
* There were no missing values or duplicate values in this table.

1. In the VENDOR table, the fields are VEND\_ID, VEND\_NAME, VEND\_STREET, VEND\_CITY, VEND\_STATE, and VEND\_ZIP.

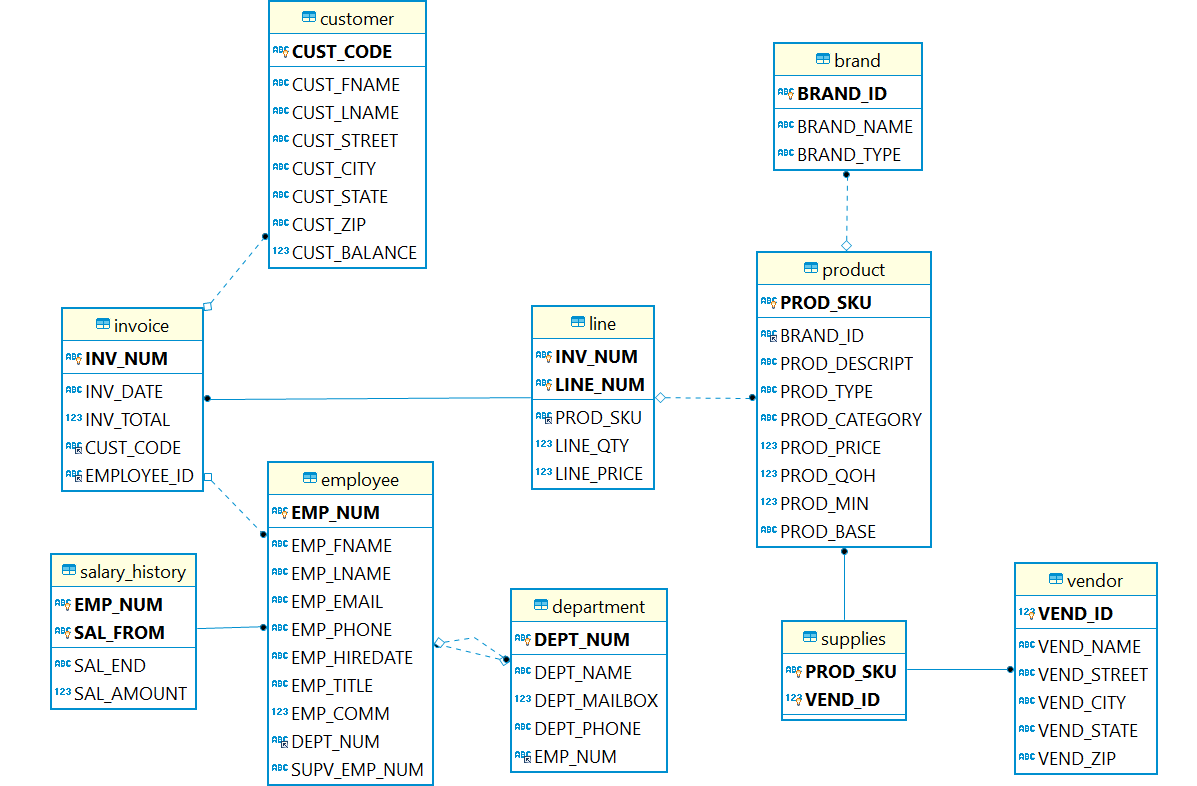
* There were some missing values in the VEND\_NAME and VEND\_STREET attributes in this table. I decided to replace missing values with reasonable logics. I noticed that the missing values have the VEND\_ID of 15, so I checked the records of VEND\_ID of 15 in the table. The figure below was what I found from the VENDOR table for VEND\_ID of 15. As a result, I replaced missing values with “Unlimited Wholesale of Ohio” for the vendor’s name and “454 WINDJAMMER CIRCLE” for vendor’s street address. I believed they were the reasonable values for the missing values.



* Next step, I grouped the data in VENDOR table by the VEND\_ID.

1. **ENTITY RELATION DIAGRAM**

After cleaning the data in ten tables, I imported all the data into MySQL using DBeaver. My next step was adding Primary Keys and Foreign Keys into all the tables. The figure below is the Entity Relation Diagram (ERD) or the snapshot of data structures I had.



1. **THE RESULTS INTERPRETED FROM THE DATA**
2. 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.

**select** e.EMP\_NUM, e.EMP\_LNAME, e.EMP\_FNAME, sh.SAL\_AMOUNT

**from** employee e **join** salary\_history sh

**on** e.EMP\_NUM = sh.EMP\_NUM

**where** sh.SAL\_END = "-99"

**and** e.DEPT\_NUM = 300

**order** **by** 4 **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.

**select** e.EMP\_NUM, e.EMP\_LNAME, e.EMP\_FNAME, sh.SAL\_AMOUNT **as** StartSalary

**from** employee e **join** salary\_history sh

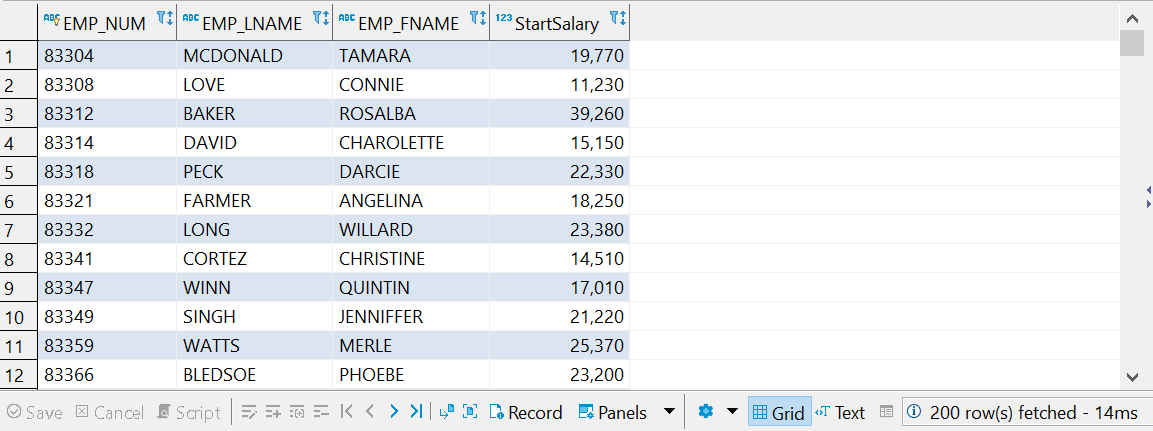
**on** e.EMP\_NUM = sh.EMP\_NUM

**where** sh.SAL\_AMOUNT = (**select** **min**(sh1.SAL\_AMOUNT)

**from** salary\_history sh1

**where** e.EMP\_NUM = sh1.EMP\_NUM)

**order** **by** 1



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.

**select** a.INV\_NUM, a.LINE\_NUM, a.PROD\_SKU, z.PROD\_DESCRIPT, z.PROD\_CATEGORY, z.BRAND\_ID **from** (

**select** l.INV\_NUM, l.LINE\_NUM, PROD\_SKU **from** line l

**where** l.PROD\_SKU

**in** (

**select** p.PROD\_SKU

**from** product p

**where** p.PROD\_CATEGORY = 'Top Coat' **or** p.PROD\_CATEGORY = 'Sealer'

)) a **join** (

**select** p.PROD\_SKU, p.PROD\_DESCRIPT, p.BRAND\_ID, p.PROD\_CATEGORY

**from** product p

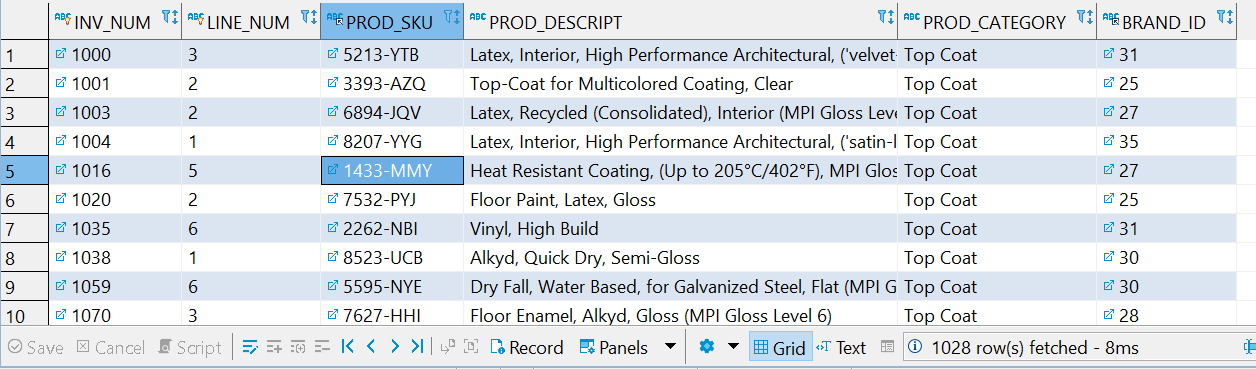
**where** p.PROD\_CATEGORY = 'Top Coat' **or** p.PROD\_CATEGORY = 'Sealer'

) z

**on** a.PROD\_SKU = z.PROD\_SKU

**order** **by** INV\_NUM

Result:



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.

In Julian date format, November 1, 2015 is 42309 and December 5, 2015 is 42343. However, in the given data, the earliest date of the INV\_DATE is 41317, and the latest date of INV\_DATE is 41655. Therefore, the time period being asked in this question is out of the time frame of the data. Therefore, I think the time for this question should be November 1, 2013 and December 5, 2013.

In Julian date format, November 1, 2013 is 41579 and December 5, 2013 is 41613.

**select** e.EMP\_NUM, e.EMP\_LNAME, e.EMP\_FNAME, **Max**(a.Total) **as** MaxTotalUnitSold

**from** employee e **join**

(**select** i.EMPLOYEE\_ID, **SUM**(l.LINE\_QTY) **as** Total

**from** invoice i **join** line l **on** i.INV\_NUM = l.INV\_NUM

**join** product p **on** l.PROD\_SKU = p.PROD\_SKU

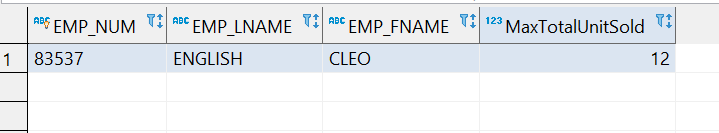
**join** brand b **on** b.BRAND\_ID = p.BRAND\_ID

**where** BRAND\_NAME = "Binder Prime"

**and** INV\_DATE **between** 41579 **and** 41613

**group** **by** EMPLOYEE\_ID) a

**on** e.EMP\_NUM = a.EMPLOYEE\_ID



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.

**select** \* **from**

(**select** c.CUST\_CODE, c.CUST\_LNAME, c.CUST\_FNAME

**from** customer c

**where** c.CUST\_CODE **in**

(**select** i.CUST\_CODE **from** invoice i **join** employee e

**on** i.EMPLOYEE\_ID = e.EMP\_NUM

**where** i.EMPLOYEE\_ID = 83649)) a

**join**

(**select** c.CUST\_CODE, c.CUST\_LNAME, c.CUST\_FNAME

**from** customer c

**where** c.CUST\_CODE **in**

(**select** i.CUST\_CODE **from** invoice i **join** employee e

**on** i.EMPLOYEE\_ID = e.EMP\_NUM

**where** i.EMPLOYEE\_ID = 83677)) a1

**where** a.CUST\_CODE = a1.CUST\_CODE

There is no result after running these queries. That means there is no customer who had both at least one invoice generated by employee with EMPLOYEE\_ID of 83649 and at least one invoice generated by employee with EMPLOYEE\_ID of 83677.

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).

**select** c.CUST\_CODE, c.CUST\_LNAME, c.CUST\_FNAME, c.CUST\_STREET, c.CUST\_CITY, c.CUST\_STATE, c.CUST\_ZIP, i.INV\_DATE, i.INV\_TOTAL **as** LargestInvoice

**from** customer c **join** invoice i **on** c.CUST\_CODE = i.CUST\_CODE

**where** c.CUST\_STATE = "AL"

**and** i.INV\_TOTAL = (**select** **Max**(i.INV\_TOTAL)

**from** invoice i1 **where** i1.CUST\_CODE = c.CUST\_CODE)

**union**

**select** c.CUST\_CODE, c.CUST\_LNAME, c.CUST\_FNAME, 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** i.CUST\_CODE **from** invoice i)

**Result:**



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.

**select** b.BRAND\_NAME, b.BRAND\_TYPE,

**round**(avgprice,2) **as** AveragePrice, unitsold **as** UnitSold

**from** brand b **join**

(**select** p.BRAND\_ID, **avg**(p.PROD\_PRICE) **as** avgprice **from** product p

**group** **by** p.BRAND\_ID) a

**on** b.BRAND\_ID = a.BRAND\_ID

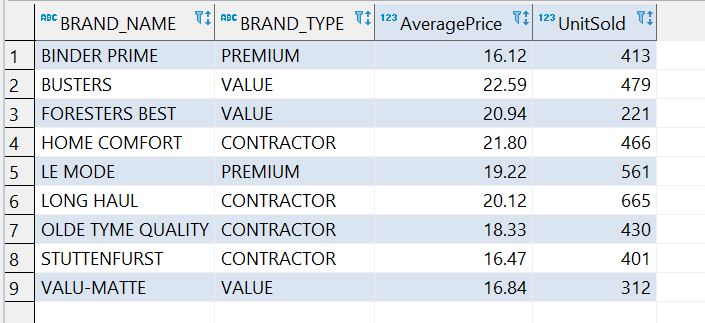
**join** (**select** p.BRAND\_ID, **sum**(l.LINE\_QTY) **as** unitsold

**from** product p **join** line l **on** p.PROD\_SKU = l.PROD\_SKU

**group** **by** p.BRAND\_ID) z

**on** b.BRAND\_ID = z.BRAND\_ID

**order** **by** 1



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.

**select** b.BRAND\_NAME, b.BRAND\_TYPE, p.PROD\_SKU, p.PROD\_DESCRIPT, p.PROD\_PRICE

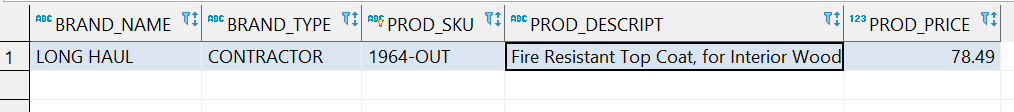
**from** brand b **join** product p **on** b.BRAND\_ID = p.BRAND\_ID

**where** b.BRAND\_TYPE <> "PREMIUM"

**and** p.PROD\_PRICE > (**select** **max**(p.PROD\_PRICE)

**from** product p **join** brand b **on** p.BRAND\_ID = b.BRAND\_ID

**where** b.BRAND\_TYPE = "PREMIUM")



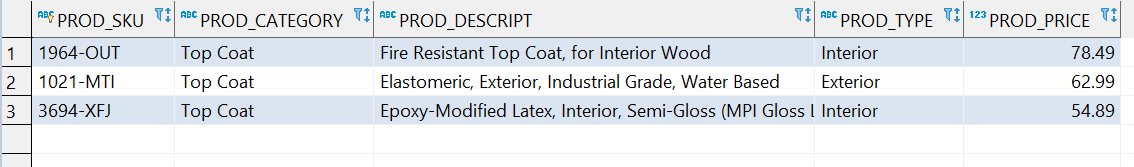
1. Using SQL descriptive statistics functions calculate the value of the following items:
   1. What are the products that have a price greater than $50?

**select** p.PROD\_SKU, p.PROD\_CATEGORY, p.PROD\_DESCRIPT, p.PROD\_TYPE, p.PROD\_PRICE

**from** product p

**where** p.PROD\_PRICE >50

**order** **by** 5 **desc**

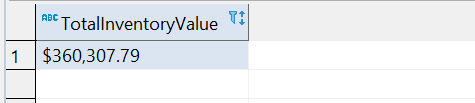


* 1. What is total value of our entire inventory on hand?

I assumed the inventory value is in dollars.

**select** **concat**('$',format(**sum**(p.PROD\_QOH \* p.PROD\_PRICE),2)) **as** TotalInventoryValue

**from** product p



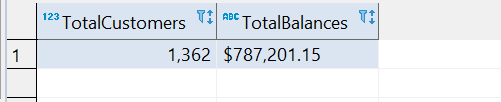
* 1. How many customers do we presently have and what is the total of all customer balances?

I assumed the customer balances are in dollars.

**select** **count**(c.CUST\_CODE) **as** TotalCustomers,

**concat**('$',format(**sum**(c.CUST\_BALANCE),2)) **as** TotalBalances

**from** customer c



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

**select** CUST\_STATE, **sum**(total) STATE\_TOTAL

**from**

(**select** c.CUST\_STATE,i.INV\_NUM

**from** invoice i **left** **join** customer c **on** i.CUST\_CODE = c.CUST\_CODE) a **join**

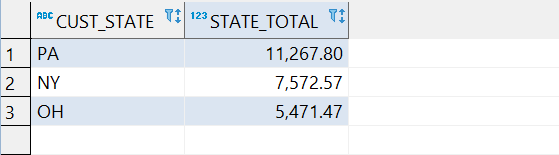
(**select** l.INV\_NUM, (l.LINE\_QTY\*l.LINE\_PRICE) total **from** line l) b

**on** a.INV\_NUM = b.INV\_NUM

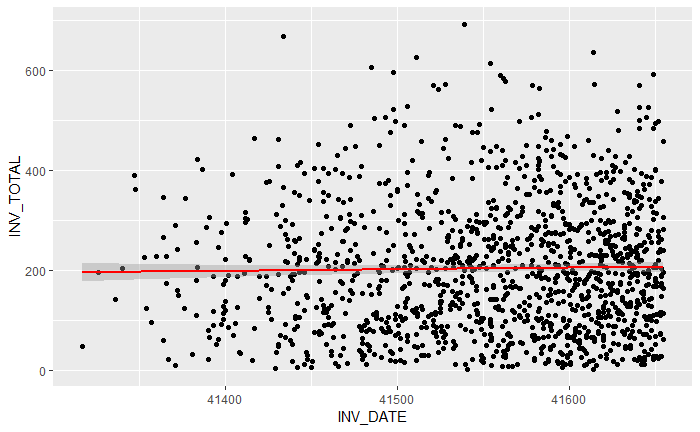
**group** **by** CUST\_STATE

**order** **by** STATE\_TOTAL **desc**

**limit** 3



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.)

I imported the data in the invoice table into R to do predictive statistics on the data. The first step was to plot the invoice total values against the dates to see the pattern of the data. From this plot, it seems that there is no specific pattern for the relationship between INV\_DATE AND INV\_TOTAL.

After plotting the data, I proceeded to perform a linear regression. The figure below shows the summary of the linear regression model. The R-squared of this model is closed to zero, and the p-value of the INV\_DATE coefficient is greater than 0.05, indicating that there is no relationship between the INV\_DATE variable and the INV\_TOTAL variable. Hence, this model is not a good model to predict the sales for the next year. As a result, I recommend not to use the INV\_DATE to forecast the sales (INV\_TOTAL).

