**Project: Company Dataset**

Contents

[ Introduction 2](#_Toc532393060)

[ Normalization and Data Cleaning 2](#_Toc532393061)

[ Logical/Physical Model Using Erwin 5](#_Toc532393062)

[ Importing the Data into MSSQL and Setting Constraints 6](#_Toc532393063)

[ SQL Queries and Results 7](#_Toc532393064)

[ Linear Regression Using R and Excel 17](#_Toc532393065)

[ Recommendations 19](#_Toc532393066)

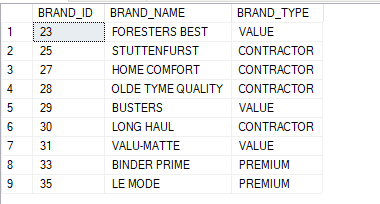
[ Conclusion 19](#_Toc532393067)

# Introduction

This project was completed as part of my required coursework as I pursue my Master’s in Business Analytics from the Jindal School of Management at UT Dallas for the course, Database Foundations for Business Analytics, with Dr. James Scott. The aim of this project was to give students exposure to a real-life application of database concepts. The problem defined by the instructor was that an organization which sold several brands of hardware materials wanted to know how best to increase its sales by looking at the relevant key performance indicators (KPIs) of high performance and investigating how to analyze them. Some specific examples of the points that the organization wanted to look into were the starting salary of employees, employees who sold the most products from a specific brand during a certain timeframe, looking at the possibility of planning an event in a certain market based on performance measures from the region, and analyzing the relationship between the time at which a product was purchased and its effect on the total sales during the time frame to make predictions regarding future performance using Linear Regression.

# Normalization and Data Cleaning

The instructor provided three datasets and a data dictionary. According to the data dictionary, students needed to derive ten tables with certain fields and constraints. In most applications of normalization in the business world, the highest normal form that needs to be achieved is 3NF. In order to normalize the datasets, I made ten different tables according to the data dictionary with their corresponding fields. As part of the normalization process, I removed repeating groups using Excel’s “Remove Duplicates” button under the Data tab. The first table, BRAND, had 252 records before normalization. When I removed repeating groups so that the final (normalized) table would have only unique values, the table had nine unique records. The attributes found in this table include BRAND\_ID, BRAND\_NAME, BRAND\_TYPE. The field, BRAND\_ID, is the unique identifier which would help point to the other fields in the table. Thus, the BRAND\_ID field can be considered to be the Primary Key for this table. The image below depicts the normalized and cleaned BRAND table.



**Figure 1: Normalized BRAND Table**

The table below shows the number of records in each table before the normalization process, the number of records in each table after the normalization process, and the primary and foreign keys for each table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | Number of Records Pre-Normalization | Number of Records Post-Normalization | Primary Key(s) | Foreign Key(s) |
| BRAND | 252 | 9 | BRAND\_ID |  |
| CUSTOMER | 1362 | 1362 | CUST\_CODE |  |
| DEPARTMENT | 6711 | 8 | DEPT\_NUM | EMP\_NUM |
| EMPLOYEE | 6711 | 363 | EMP\_NUM | DEPT\_NUM |
| INVOICE | 1362 | 1347 | INV\_NUM | CUST\_CODE  EMPLOYEE\_ID |
| LINE | 1362 | 1347 | LINE\_NUM  INV\_NUM | INV\_NUM  PROD\_SKU |
| PRODUCT | 252 | 252 | PROD\_SKU | BRAND\_ID |
| SALARY\_HISTORY | 6710 | 6710 | SAL\_FROM  EMP\_NUM | EMP\_NUM |
| SUPPLIES | 252 | 252 | PROD\_SKU  VEND\_ID | PROD\_SKU  VEND\_ID |
| VENDOR | 252 | 22 | VEND\_ID |  |

**Table 1: Tables with Pre-normalization and Post-normalization records, Primary Keys, and Foreign Keys**

During the normalization process, I noticed that there were repeated values in the INV\_NUM field in the INVOICE table had the same invoice number for different customers. This is an impossible condition. In order to ensure that INV\_NUM met the integrity constraints set by the DBMS so as to be considered a primary key for the INVOICE table, I removed the records below from the INVOICE table.

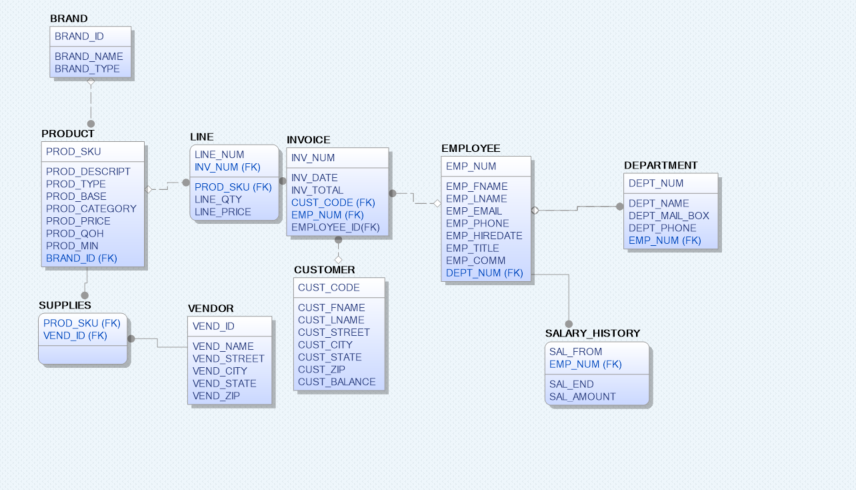
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INV\_NUM | INV\_DATE | CUST\_CODE | INV\_TOTAL | EMPLOYEE\_ID |
| 2275 | 41532 | 393 | 158.95 | 84106 |
| 2577 | 41564 | 729 | 72.75 | 84106 |
| 1978 | 41500 | 850 | 203.82 | 83677 |
| 2364 | 41542 | 877 | 300.32 | 83906 |
| 2921 | 41599 | 888 | 69.87 | 83993 |
| 2124 | 41516 | 917 | 172.91 | 83906 |
| 3370 | 41646 | 928 | 129.93 | 84248 |
| 2644 | 41571 | 989 | 108.13 | 84078 |
| 2315 | 41536 | 993 | 124.24 | 84163 |
| 2362 | 41542 | 1014 | 212.03 | 83537 |
| 3024 | 41610 | 1207 | 181.92 | 83705 |
| 2885 | 41596 | 1218 | 7.49 | 83850 |
| 3135 | 41622 | 1307 | 25.37 | 83649 |
| 3347 | 41643 | 1460 | 195.47 | 84021 |
| 3364 | 41645 | 1410 | 72.94 | 83734 |

**Table 2: Records Removed from the INVOICE Table**

Since the LINE table also has the INV\_NUM field, the same process needed to be executed during the normalization process in this table too.

The Data Cleaning process involved removing special characters from the data, filling in missing values, and changing the format of records as necessary. The first three tables did not require any data cleaning. I did a visual check of the data to confirm this. In the EMPLOYEE table, I changed the date format for the EMP\_HIREDATE field from the number of days since 1/1/1900 to MSSQL date format (i.e. 1900-01-01). I followed the same procedure for dates in the INVOICE and SALARY\_HISTORY tables. The PRODUCT table had some discrepancies in the PROD\_DESCRIPT field wherein some records had descriptions which read Light Industrial Coating, Exterior, Water Based ((Eggshell-like) - MPI Gloss Level 3) while other records in the same field read Light Industrial Coating, Exterior, Water Based ('eggshell-like' - MPI Gloss Level 3). In order to remove this discrepancy, I changed all records which had phrases like ‘eggshell-like’, ‘satin-like’, or ‘velvet-like’ to (Eggshell-like), (Satin-like), and (Velvet-like) respectively. I noticed that there were 6 records in the PROD\_SKU and PROD\_QOH fields in the PRODUCT table that contained special characters. I edited these records accordingly. In the SALARY\_HISTORY table’s SAL\_END field, several records contained ‘-’ to indicate that the salary was current for the corresponding employee. In order to make this more readable for the DBMS, I changed the ‘-’ to NULL. In the SUPPLIES table too, I found six records with special characters in the PROD\_SKU field. The VENDOR table, after normalization, had 23 records. However, one record of the 23 had missing values for VEND\_NAME, VEND\_STREET, VEND\_CITY, VEND\_STATE, AND VEND\_ZIP. I looked at VEND\_ID and noticed that it was the same as the VEND\_ID for another record (15). Because the other fields would be determined by the VEND\_ID, I deleted the 23rd record and had a final normalized and cleaned VENDOR table with 22 records.

# Logical/Physical Model Using Erwin



**Figure 2: Logical/Physical Model Using Erwin**

The image above shows the ten tables with their fields and primary key and foreign key constraints. I used the Erwin Data Modeler to generate this model. The Model depicts the relationships between the 10 entities. During the process of relationship generation, the foreign keys for each table are set. In the image above, notice the INVOICE table. The INVOICE table is supposed to have an attribute named EMPLOYEE\_ID, which is for those employees who are commissioned. In essence, this field also depends on the EMP\_NUM field because it is part of employees, the only caveat is that it points to a very specific group of employees.

# Importing the Data into MSSQL and Setting Constraints

Once the data tables were normalized and cleaned, I loaded them into the MSSQL Management Studio. I imported the data into SQL using built-in functionalities. After loading the data, I set Primary Key and Foreign Key constraints on each table using the code in the following table.

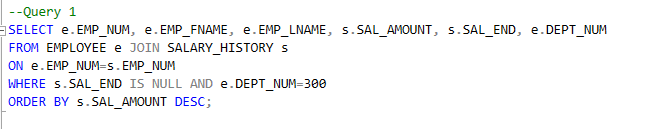
|  |  |  |
| --- | --- | --- |
| Table Name | Primary Key | Foreign Key |
| BRAND | ALTER TABLE BRAND  ADD PRIMARY KEY(BRAND\_ID); |  |
| CUSTOMER | ALTER TABLE CUSTOMER  ADD PRIMARY KEY (CUST\_CODE); |  |
| DEPARTMENT | ALTER TABLE DEPARTMENT  ADD PRIMARY KEY(DEPT\_NUM); | ALTER TABLE DEPARTMENT  ADD FOREIGN KEY (EMP\_NUM) REFERENCES EMPLOYEE (EMP\_NUM); |
| EMPLOYEE | ALTER TABLE EMPLOYEE  ADD PRIMARY KEY (EMP\_NUM); | ALTER TABLE EMPLOYEE  ADD FOREIGN KEY (DEPT\_NUM)  REFERENCES DEPARMENT (DEPT\_NUM); |
| INVOICE | ALTER TABLE INVOICE  ADD PRIMARY KEY (INV\_NUM); | ALTER TABLE INVOICE  ADD FOREIGN KEY (CUST\_CODE)  REFERENCES CUSTOMER (CUST\_CODE),  FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEE (EMPLOYEE\_ID); |
| LINE | ALTER TABLE LINE  ADD PRIMARY KEY (INV\_NUM),  PRIMARY KEY (LINE\_NUM); | ALTER TABL LINE  ADD FOREIGN KEY (INV\_NUM) REFERENCES INVOICE (INV\_NUM),  FOREIGN KEY (PROD\_SKU) REFERENCES PRODUCT (PROD\_SKU); |
| PRODUCT | ALTER TABLE PRODUCT  ADD PRIMARY KEY (PROD\_SKU); | ALTER TABLE PRODUCT  ADD FOREIGN KEY (BRAND\_ID) REFERENCES BRAND (BRAND\_ID) |
| SALARY\_HISTORY | ALTER TABLE SALARY\_HISTORY  ADD PRIMARY KEY (SAL\_END),  PRIMARY KEY (EMP\_NUM); | ALTER TABLE SALARY\_HISTORY  ADD FOREIGN KEY (EMP\_NUM) REFERENCES EMPLOYEE (EMP\_NUM) |
| SUPPLIES | ALTER TABLE SUPPLIES  ADD PRIMARY KEY (PROD\_SKU),  PRIMARY KEY (VEND\_ID); | ALTER TABLE SUPPLIES  ADD FOREIGN KEY (PROD\_SKU) REFERENCES PRODUCT (PROD\_SKU),  FOREIGN KEY (VEND\_ID) REFERENCES VENDOR (VEND\_ID); |
| VENDOR | ALTER TABLE VENDOR  ADD PRIMARY KEY(VEND\_ID); |  |

**Table 3: Primary Key and Foreign Key Constraints**

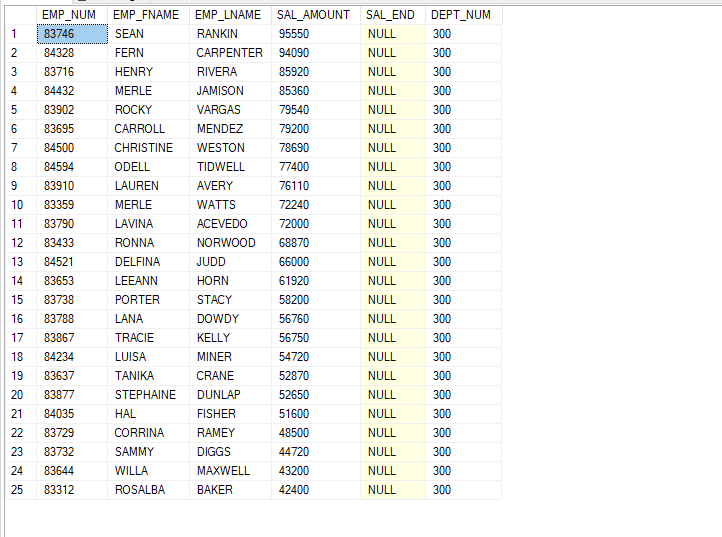
# SQL Queries and Results

The SQL Queries for this project were programmed with the aim of answering questions as set by the instructor.

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.

****

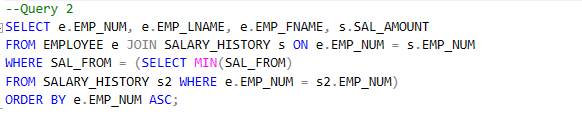
**Figure 3: Query 1 For Question 1**



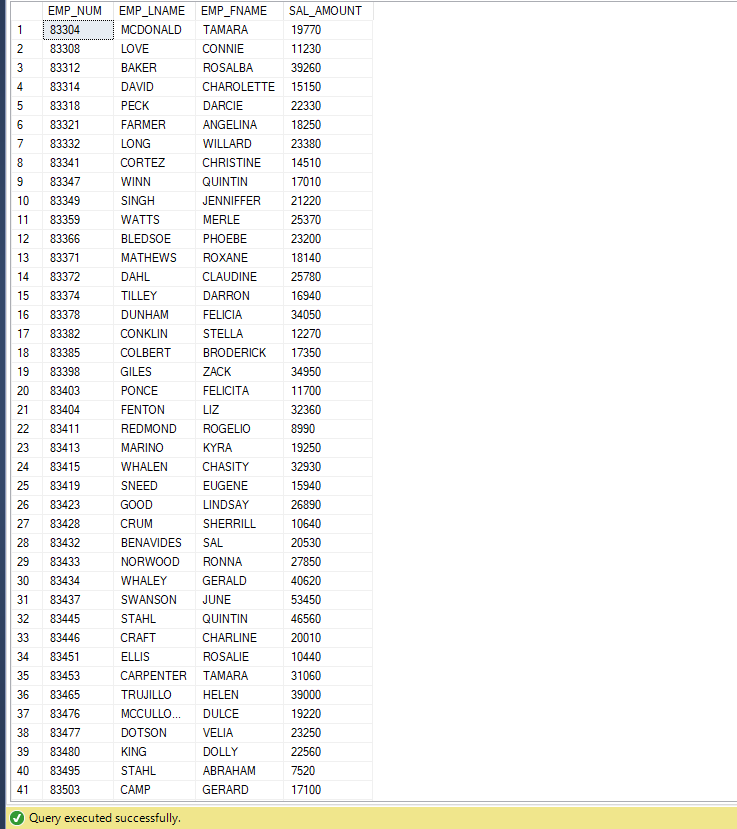
**Figure 4: Query 1 Results**

The output this query yields is that there are 25 cases wherein the employee’s department is 300 and they are current employees (no salary end date). The highest salary earned in this group is $95550 and the lowest salary earned in this group is $42400. Here, I have used a JOIN to get information from the EMPLOYEE and SALARY\_HISTORY tables for the final result set.

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.



**Figure 5: Query 2 for Question 2**



**Figure 6: Query 2 Result**

The image above shows a small sampling of the result set generated by Query 2. The actual result set returns 363 records that meet the requirement of generating the starting salary for each employee. Here, I have used a JOIN to get information from the EMPLOYEE and SALARY\_HISTORY tables and the SELECT MIN subquery to get the starting salary for each employee.

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.

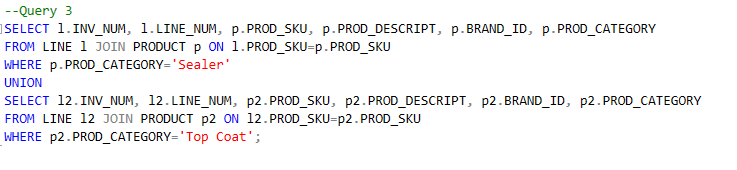
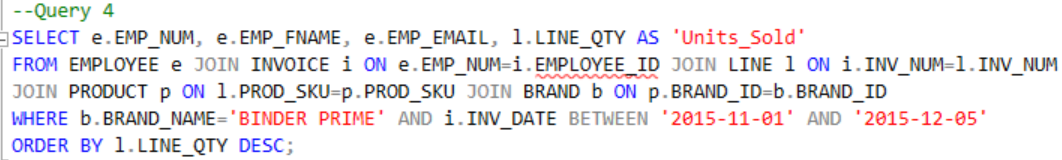




Figure 7: Query 3 for Question 3

Figure 8: Query 3 Results

The result set above depicts 25 records out of 3453 records. These records meet the requirement of being from either the ‘Sealer’ or ‘Top Coat’ product category. Here, I have used JOIN to join tables in order to get information from both the LINE table and the PRODUCT table. Since the result set required both products from the Sealer category and the Top Coat category, I used UNION to get both the result sets using one query.

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.

**Figure 9: Query 4 for Question 4**



**Figure 10: Result Set for Query 4**

The query above does not generate a result set because there are no invoice dates between November 1, 2015 and December 5, 2015 in the dataset. I did a visual check of the dataset to confirm this. In order to produce the result set, I used several joins between tables to connect the Employees with Invoices, product quantities and the brand, ‘Binder Prime’.

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.

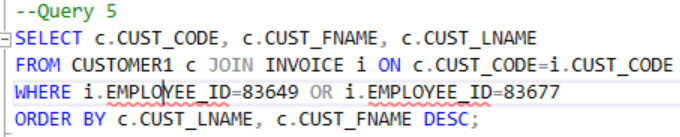
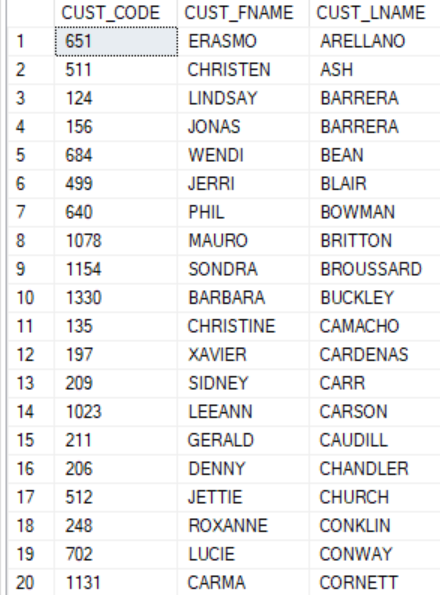
****

Figure 11: Query 5 for Question 5

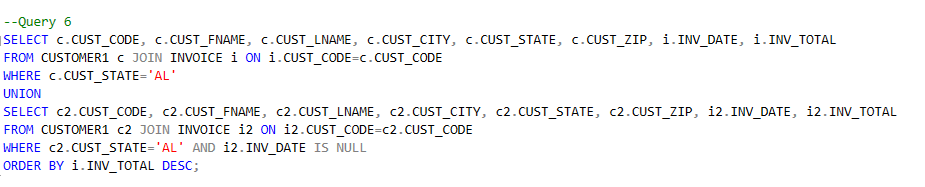
****

**Figure 11: Query 5 for Question 5**

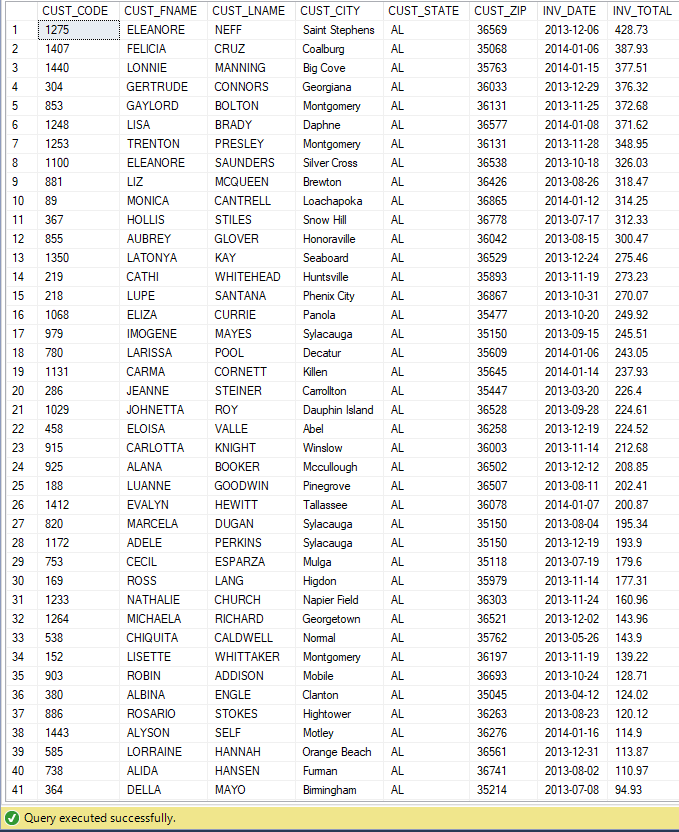
**Figure 12: Result Set for Query 5**

The result set above is part of a result set of 97 records. All the customers in this situation had invoices completed by employees with EMPLOYEE\_ID=83649 or 83677. Here, I used a JOIN on the CUSTOMER and INVOICE tables in order to get information from both tables to generate the result set as stipulated in the question.

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).

****

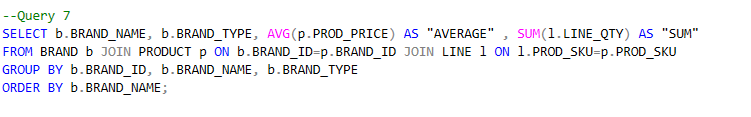
**Figure 13: Query 6 for Question 6**

****

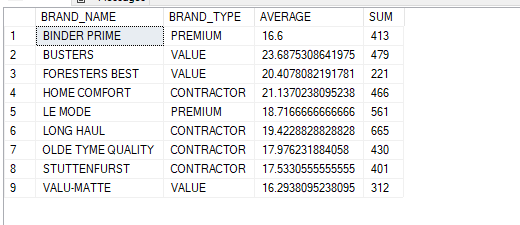
**Figure 14: Result Set for Query 6**

The image above shows part of the result set generated for the largest purchases made by customers in Alabama. The actual result set has 49 records. Here, I have used the JOIN operator to get information from both the CUSTOMER and INVOICE tables. I also used the UNION operator so that I could cater for both customers who had made purchases and those who had not.

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.

****

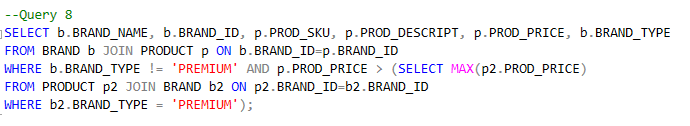
**Figure 15: Query 7 for Question 7**

****

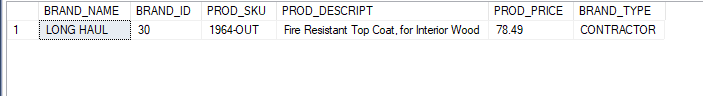
**Figure 16: Result set for Query 7**

The result set above consists of nine records. Each record has the name of the brand, the type of brand, the average price of products from the brand, and the total number of units sold of the brand. I used a JOIN operator to get information from both the BRAND and PRODUCT tables. I used GROUP BY to collect information regarding the average price of products and the total number of units sold by brand in order to organize the result set better.

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.



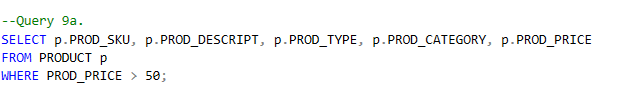
**Figure 17: Query 8 for Question 8**

****

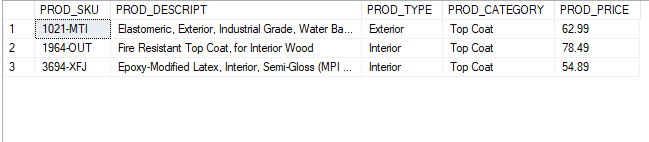
**Figure 18: Result Set for Query 8**

The result set above shows the brand name, brand ID, product sku, product description, the product price, and brand type for products that meet the requirement of not being premium and costing more than premium products. This generates one result set. Here, I have used JOIN and the SELECT MAX subquery in order to generate products which are not of type premium and are more expensive than the most expensive premium products.

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?



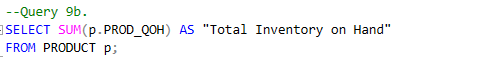
**Figure 19: Query 9a. for Question 9a.**



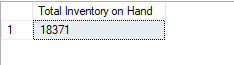
**Figure 20: Result Set for Query 9a.**

The result set above shows those products whose prices are greater than $50. Here, I have used Basic SQL functionality to generate the above result set.

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



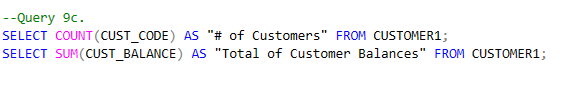
**Figure 21: Query 9b for Question 9b**



**Figure 22: Result Set for Query 9b**

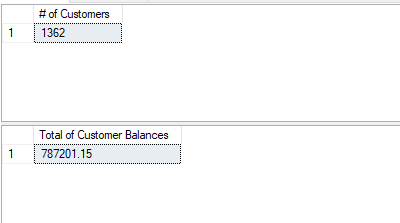
The total value of the entire inventory on hand is 18,371. Here, I have used the SUM function to generate the total inventory on hand for the organization.

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



**Figure 23: Query 9c. for Question 9c**

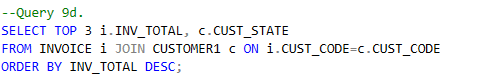
2



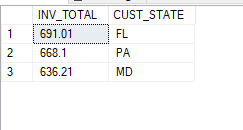
**Figure 24: Result Set for Query 9c**

The organization has 1362 customers at present and the total of customer balances is $787,201.15. Here, I used the COUNT and SUM SQL functions respectively to generate the total number of customers that the organization has and the total of all the customers’ balances.

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



**Figure 25: Query 9d for Question 9d**

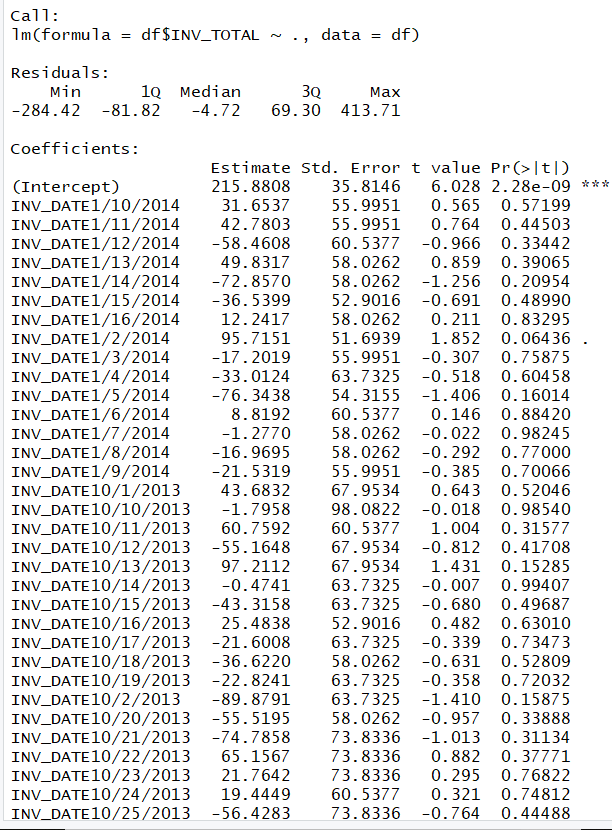


**Figure 26: Result Set for Query 9d**

The top three states that buy the most products from the organization are Florida, Pennsylvania, and Maryland. Here, I used the TOP 3 function from SQL to generate the three states that buy the most products in dollars from the organization.

# Linear Regression Using R and Excel

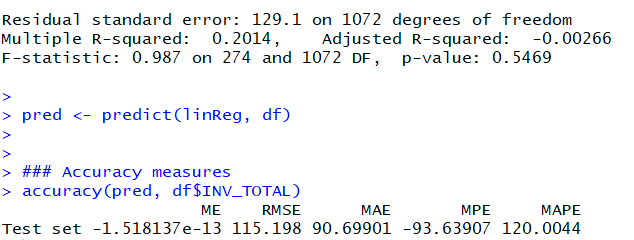
I performed linear regression analysis using INV\_TOTAL as the dependent variable and INV\_DATE as the independent variable using both Excel and R.



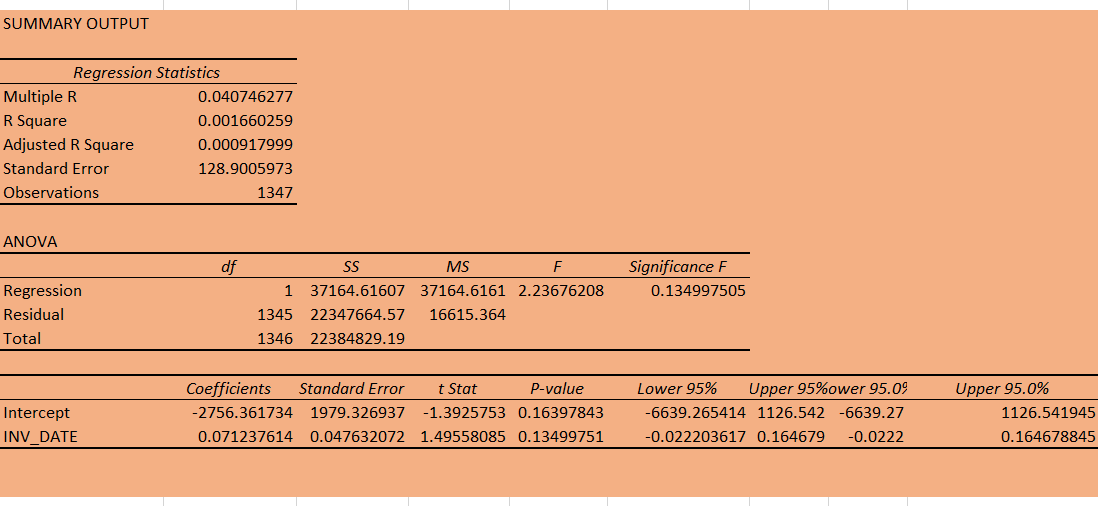
**Figure 27: Result from Regression in R**

Here, we see the coefficients for each date. The values of these coefficients are fairly large and fairly similar to one another. The significance level of the intercept shows that the intercept is the only factor which can considered valuable for prediction purposes. Unfortunately, the predictor variable cannot be considered a good predictor for INV\_TOTAL. The image below in Figure 25 shows the values for R2, adjusted R2, and the p-value. The value of the adjusted R-squared is negative and fairly close to 0. This means that model is not improved as such by linear regression. The p-value for the model is quite large at 0.5469.

The diagram in Figure 26 shows the results from regression in Excel. This was performed with a 95% confidence interval. The coefficient for INV\_DATE is 0.07 here and the adjusted R-squared value is 0.0009. This yields the same interpretation that the model does not perform very well as far as prediction of INV\_TOTAL is concerned.

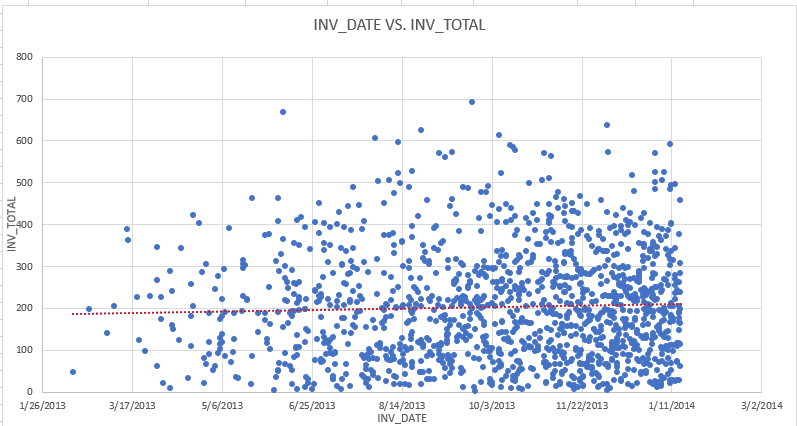


**Figure 28: Linear Regression in R**



**Figure 29: Result from Regression in Excel**

The graph below in Figure 27 shows that there INV\_TOTAL values don’t generally increase as the INV\_DATE values increase.



**Figure 30: Graph for Regression in Excel**

# Recommendations

Based on the information gleaned from running the queries, I would recommend that the organization focus their business on Florida, Pennsylvania, and Maryland because these three states are the ones which spend the most dollars on products. I would also recommend that the organization plan the promotion in Alabama because customers in that state have high invoice totals. I would also recommend including fields like product name in the PRODUCT table because that would help make the process more intuitive. I would also recommend have more dates for invoice in the dataset so that the spread is better.

# Conclusion

In conclusion, working on this project was a fruitful experience because it gave me experience of working on a real-life application of implementing a DBMS. Working on this project has also given me experience in applying my knowledge of normalization and data cleaning. Through this project, I learned to convert raw data into something meaningful through normalization and then to import it into MS SQL. I also learned to implement a DBMS by setting Primary Key and Foreign Key constraints. This project also helped me tie in what I learned in my Business Analytics with R course through the exercise on performing Linear Regression on the Invoice Date and Invoice Number.