
INDIVIDUAL PROJECT DATABASE FOUNDATIONS

INVENTORY MODEL DESIGN DATABASE

PHANINDRA PANTHAGANI

NETID: PXP180008

Assumptions and Data Cleaning Part +Normalization and Data Modelling

1) DATA MUNGING:

Each of these tables were then created according to the fields using R and selecting particular fields into particular entities. There were many duplicates for each of the tables and duplicates were removed using Remove Duplicates in Excel. The R code used is as below:

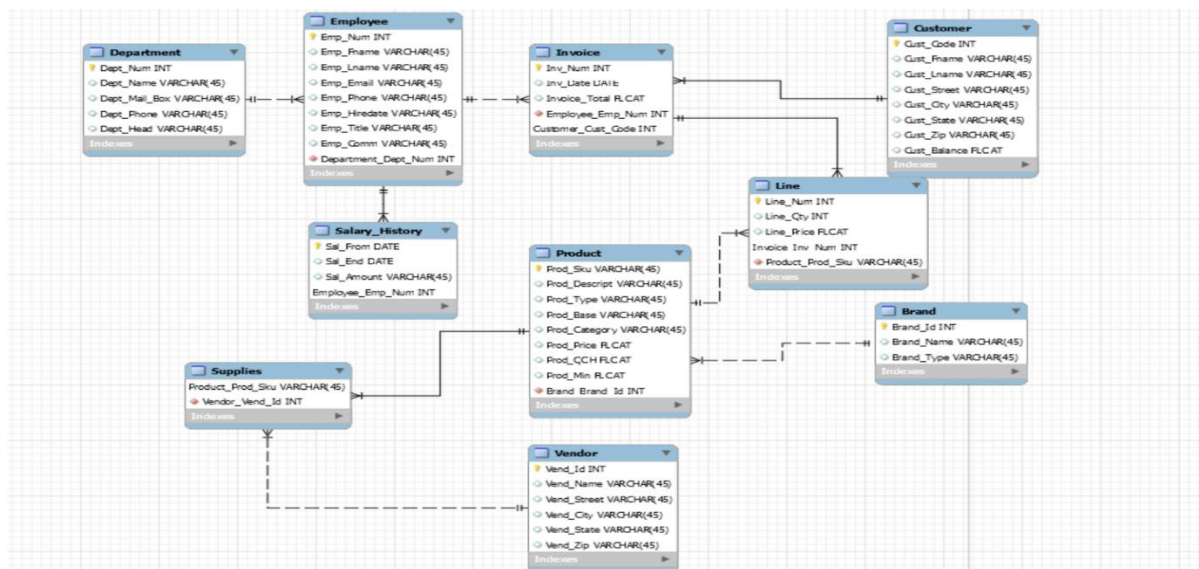
There were many special characters like \$,~ etc and NULL Values which were present. The Special Characters were removed and the rows containing the NULL Values were completely removed except in Salary_History Table.

2) NORMALIZATION:

I looked at the data and started Normalizing the data, but I faced a lot of problems while trying to do it. So the first thing I did was to visualize the whole data by preparing the data model and then choosing the appropriate Primary & Foreign Keys.

3) DATA MODEL

The data model looked as below:



4) SALARY_HISTORY TABLE FORMATTING

The Salary_History table had date in the wrong format by default like mm/dd/yyyy . It was changed to the proper MySQL format which is (YYYY-DD-MM) which was done in excel. For this I used the DATEVALUE function(Julian Date) in Excel and then using Custom Format to change the date to YYYY-DD-MM. A snapshot is shown below.

EMP_NUM	SAL_FROM	SAL_END	SAL_AMOUNT	FORMATTEDDATE	DATETIMEVALUESOFSAL_END
84583	01/01/1985	01/01/1986	46000	1985-01-01	=DATEVALUE(C2)
83723	01/01/1993	01/01/1994	18750	1993-01-01	
83650	01/02/1999	01/01/2000	24090	1999-01-02	
84007	01/02/2000	01/01/2001	15170	2000-01-02	
83650	01/01/2001	01/01/2002	26050	2001-01-01	
84007	01/03/2002	01/01/2003	16090	2002-01-03	
84007	01/02/2003	01/01/2004	16570	2003-01-02	
83349	01/02/2004	01/01/2005	43170	2004-01-02	
84007	01/01/2005	01/01/2006	17580	2005-01-01	
83349	01/03/2006	01/01/2007	46680	2006-01-03	
83349	01/02/2007	01/01/2008	49080	2007-01-02	
83593	01/02/2008	01/01/2009	97340	2008-01-02	
83349	01/01/2009	01/01/2010	52000	2009-01-01	
83593	01/03/2010	01/01/2011	105270	2010-01-03	
83593	01/02/2011	01/01/2012	108430	2011-01-02	
83593	01/01/2013	01/01/2014	117270	2013-01-01	
84007	01/03/1996	01/02/1997	11810	1996-01-03	
84007	01/04/1998	01/02/1999	12770	1998-01-04	
84007	01/03/1999	01/02/2000	14050	1999-01-03	
83349	01/03/2000	01/02/2001	35870	2000-01-03	
84007	01/02/2001	01/02/2002	15620	2001-01-02	
83349	01/04/2002	01/02/2003	38060	2002-01-04	
83349	01/03/2003	01/02/2004	41110	2003-01-03	
83593	01/03/2004	01/02/2005	89480	2004-01-03	

DEALING WITH NULL VALUES IN SALARY_HISTORY:

All NULL Values in Salary_History were replaced with 1999-09-12 this date because MySQL doesn't import NULL Values and this was used in the Queries. For example in the first Query this date was used for Sal_End=1999-09-12

QUESTIONS:

What to Do:

1. Start understanding the data by normalizing the provided datasets using the Data Dictionary.

Remember that this takes up to 80% of your time in completing this project, so start early.

Ans: The dataset was imported in R and then exported into excel after forming the 10 different tables as given in the document. The R code is present at the end of the document named as SPLITTING INTO DIFFERENT TABLES R CODE :

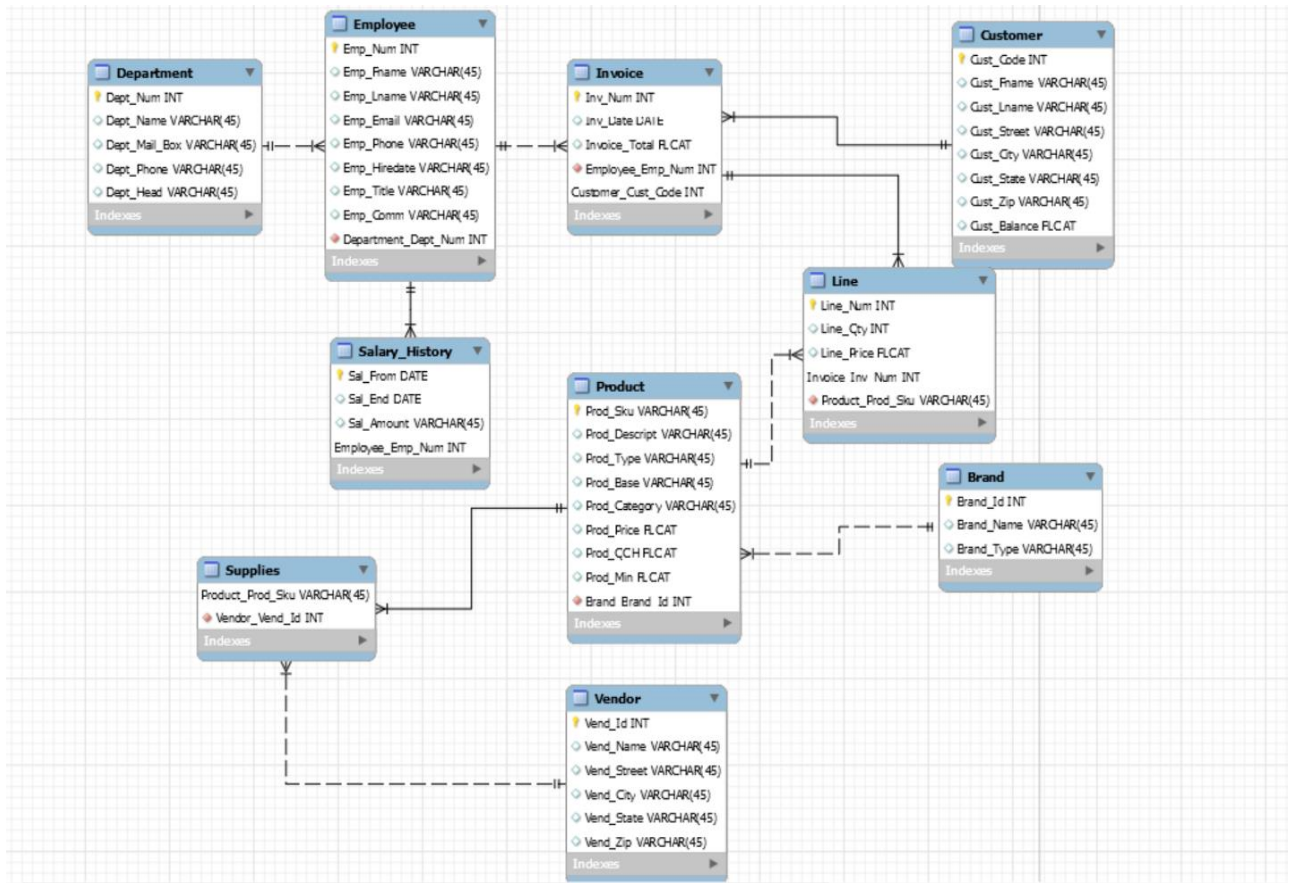
I looked at the data and started Normalizing the data, but I faced a lot of problems while trying to do it. So the first thing I did was to visualize the whole data by preparing the data model and then choosing the appropriate Primary & Foreign Keys.

- Then clean (data munging) each of the corresponding tables, inserting flags, removing bad characters, and filling in missing data fields with known values where appropriate.

Ans: The data cleaning was done by importing the data in R and then exporting into excel. Foreign Characters like?, ~ etc. were removed and data was cleaned. The R file is attached along with the Report.

- Create a Logical and Physical Model using Erwin or DBeaver.

Ans: Logical & Physical model is as below in the next page:



- Implement in your choice of DBMS (MySQL, MSSQL, or Oracle) (Please remember that Oracle Online has a table size limitation) by importing the cleaned data. Don't forget to place all required CONSTRAINTS, FOREIGN KEYS, and PRIMARY KEYS.

Ans: The DBMS used was MySQL. Forward engineering was done after creating the ER diagram using data model. CONSTRAINTS, FOREIGN KEYS, and PRIMARY KEYS were implemented in the data model.

- 5. Write SQL Queries to produce the results in answering the below questions. Include both the SQL Query code and output results in your final report.**
6. Perform a linear regression using R or other preferred statistical application. Interpret the regression analysis to arrive at a suitable forecast.
Multiple models with AIC & BIC.
7. Write and submit a report of your findings with all diagrams, graphs, output, and recommendations through eLearning Turnitin by Dec 7, 2018 at 11:59 PM.

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.

Ans:

Query:

```
SELECT      S.Employee_Emp_Num      as      EMP_ID,E.Emp_Fname      as
First_Name,E.Emp_Lname as Last_name,S.Sal_Amount as Current_Salary
FROM salary_history S INNER JOIN employee E
ON S.Employee_Emp_Num=E.Emp_Num
WHERE S.Sal_End = '1999-09-12' AND E.Department_Dept_Num=300 ORDER BY
Sal_Amount DESC;
```

Ouptut :

The screenshot shows the SQL Developer interface. The left pane displays the 'Schemas' tree with 'salary_history' selected. The main pane shows a SQL query: `SELECT S.Employee_Emp_Num as EMP_ID,E.Emp_Fname as First_Name,E.Emp_Lname as Last_name,S.Sal_Amount as Current_Salary FROM salary_history S INNER JOIN employee E ON S.Employee_Emp_Num=E.Emp_Num WHERE S.Sal_End < '1999-09-12' AND E.Department_Dept_Num=300 ORDER BY Sal_Amount DESC;`. The 'Result Grid' shows 25 rows of data. The bottom pane shows the 'Output' window with a list of actions and their durations.

EMP_ID	First_Name	Last_name	Current_Salary
83746	SEAN	RANKIN	95550
84328	FERN	CARPENTER	94090
83716	HENRY	RIVERA	85920
84432	MERLE	JAMISON	85360
83902	ROCKY	VARGAS	79540
83695	CARROLL	MENDEZ	79200
84500	CHRISTINE	WESTON	78690
84594	ODELL	TIDWELL	77400
83910	LAUREN	AVERY	76110
83359	MERLE	WATTS	72240
83790	LAVINA	ACEVEDO	72000
83433	RONNA	NORWOOD	68870
84521	DELFINA	JUDD	66000
83653	LEEANN	HORN	61920
83738	PORTER	STACY	58200
83788	LANA	DOWDY	56760
83867	TRACIE	KELLY	56750
84234	LUISA	MINER	54720
83637	TANIKA	CRANE	52870
83877	STEPHAINE	DUNLAP	52650
84035	HAL	FISHER	51600
83729	CORRINA	RAMEY	48500
83732	SAMMY	DIGGS	44720

Since its only 25 rows pasting the Output.

EMP_ID	First_Name	Last_name	Current_Salary
83746	SEAN	RANKIN	95550
84328	FERN	CARPENTER	94090
83716	HENRY	RIVERA	85920
84432	MERLE	JAMISON	85360
83902	ROCKY	VARGAS	79540
83695	CARROLL	MENDEZ	79200
84500	CHRISTINE	WESTON	78690
84594	ODELL	TIDWELL	77400
83910	LAUREN	AVERY	76110
83359	MERLE	WATTS	72240
83790	LAVINA	ACEVEDO	72000
83433	RONNA	NORWOOD	68870
84521	DELFINA	JUDD	66000
83653	LEEANN	HORN	61920
83738	PORTER	STACY	58200
83788	LANA	DOWDY	56760
83867	TRACIE	KELLY	56750
84234	LUISA	MINER	54720
83637	TANIKA	CRANE	52870
83877	STEPHAINE	DUNLAP	52650
84035	HAL	FISHER	51600
83729	CORRINA	RAMEY	48500
83732	SAMMY	DIGGS	44720

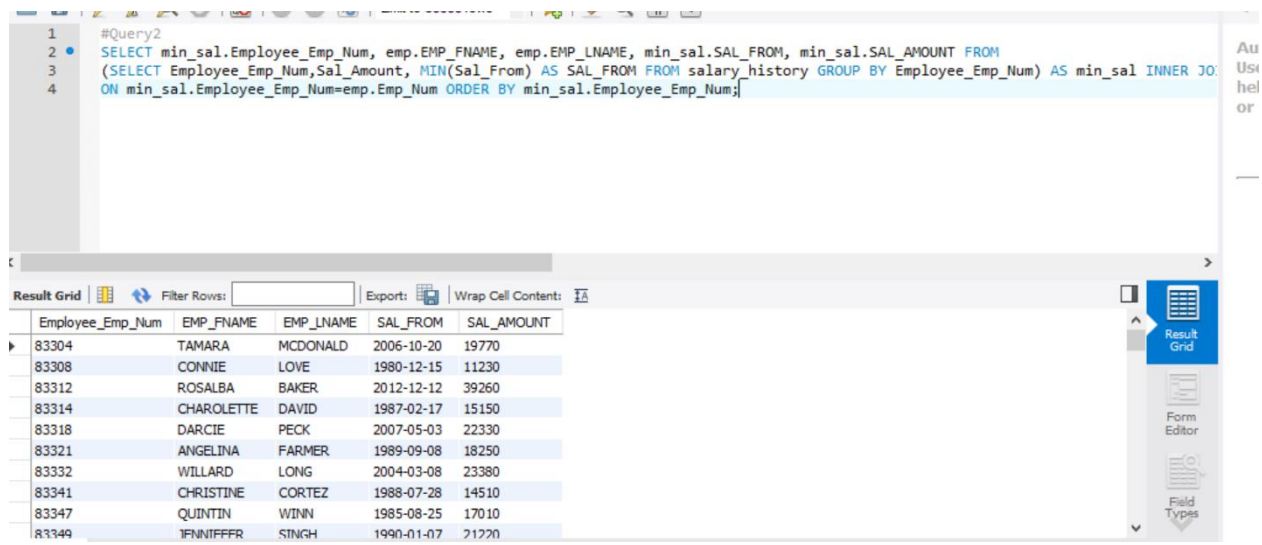
83644	WILLA	MAXWELL	43200
83312	ROSALBA	BAKER	42400

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 min_sal.Employee_Emp_Num, emp.EMP_FNAME, emp.EMP_LNAME,
min_sal.SAL_FROM, min_sal.SAL_AMOUNT FROM
(SELECT Employee_Emp_Num,Sal_Amount, MIN(Sal_From) AS SAL_FROM
FROM salary_history GROUP BY Employee_Emp_Num) AS min_sal INNER JOIN
employee emp
ON min_sal.Employee_Emp_Num=emp.Emp_Num
ORDER BY min_sal.Employee_Emp_Num;
```

Ouptut:



The screenshot shows a database query editor with a query window and a result grid. The query window contains the following SQL code:

```
#Query2
1 SELECT min_sal.Employee_Emp_Num, emp.EMP_FNAME, emp.EMP_LNAME, min_sal.SAL_FROM, min_sal.SAL_AMOUNT FROM
2 (SELECT Employee_Emp_Num,Sal_Amount, MIN(Sal_From) AS SAL_FROM FROM salary_history GROUP BY Employee_Emp_Num) AS min_sal INNER JO
3 ON min_sal.Employee_Emp_Num=emp.Emp_Num ORDER BY min_sal.Employee_Emp_Num;
```

The result grid displays the following data:

Employee_Emp_Num	EMP_FNAME	EMP_LNAME	SAL_FROM	SAL_AMOUNT
83304	TAMARA	MCDONALD	2006-10-20	19770
83308	CONNIE	LOVE	1980-12-15	11230
83312	ROSALBA	BAKER	2012-12-12	39260
83314	CHAROLETTE	DAVID	1987-02-17	15150
83318	DARCIE	PECK	2007-05-03	22330
83321	ANGELINA	FARMER	1989-09-08	18250
83332	WILLARD	LONG	2004-03-08	23380
83341	CHRISTINE	CORTEZ	1988-07-28	14510
83347	QUINTIN	WINN	1985-08-25	17010
83349	IFNINIFFER	SINGH	1990-01-07	21220

363 rows were returned.They will be attached as Query2Output.CSV

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      line.Invoice_Inv_Num,      line.LINE_NUM,      line.Product_Prod_Sku,
p1.PROD_DESCRIPT,p1.Brand_Brand_Id
FROM      line      line      INNER      JOIN(SELECT      DISTINCT
p.PROD_SKU,p.PROD_DESCRIPT,p.Brand_Brand_Id      FROM      product      p      WHERE
p.PROD_CATEGORY = 'Top Coat') p1
ON      line.Product_Prod_Sku=p1.PROD_SKU      INNER      JOIN(SELECT      DISTINCT
p.PROD_SKU,p.PROD_DESCRIPT,p.Brand_Brand_Id      FROM      product      p      WHERE
p.PROD_CATEGORY = 'Sealer') p2
ON      line.Product_Prod_Sku=p2.PROD_SKU      WHERE
p1.Brand_Brand_Id=p2.Brand_Brand_Id;
```

Output:

0 rows were returned. I went back and cross check if there any sealer and top coat products of the same brand on the same invoice.

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 emp_num, First_name, LSt_name, email, tot_q FROM (SELECT SUM(l1.Line_Qty)
AS tot_q ,emp1.Emp_Num AS emp_num, emp1.EMP_FNAME AS First_name,
emp1.EMP_LNAME AS LSt_name, emp1.EMP_EMAIL AS email FROM line l1 INNER
JOIN invoice in1
ON l1.Invoice_Inv_Num=in1.Inv_Num INNER JOIN employee emp1
ON in1.Employee_Emp_Num=emp1.Emp_Num INNER JOIN product p1
ON l1.Product_Prod_Sku=p1.PROD_SKU INNER JOIN brAND b1
ON b1.Brand_Id=p1.Brand_Brand_Id WHERE b1.BRAND_NAME = 'BINDER PRIME' AND
in1.Inv_Num not like '-%')
GROUP BY emp1.Emp_Num) q1 WHERE tot_q = (SELECT MAX(abc1.tot_q) FROM
(SELECT SUM(l1.Line_Qty) AS tot_q ,emp1.Emp_Num AS emp_num,
emp1.EMP_FNAME AS First_name, emp1.EMP_LNAME AS LSt_name, emp1.EMP_EMAIL
AS email FROM line l1 INNER JOIN invoice in1
ON l1.Invoice_Inv_Num=in1.Inv_Num INNER JOIN employee emp1
ON in1.Employee_Emp_Num=emp1.Emp_Num INNER JOIN product p1 ON
l1.Product_Prod_Sku=p1.PROD_SKU INNER JOIN brAND b1
ON b1.Brand_Id=p1.Brand_Brand_Id
WHERE b1.BRAND_NAME = 'BINDER PRIME' AND in1.Inv_Num not like '-%'
AND in1.INV_DATE between ('2015-11-01') AND ('2015-12-05') GROUP BY emp1.Emp_Num)
abc1)
ORDER BY LSt_name;
```

Output:

0 rows

We see from INVOICE table that there is no invoice for the year 2015 and that's why we are getting empty output.

To check if the Query is right, I removed the inv.INV_DATE between part and checked if the query is returning any rows. I could get rows returned when I Removed the INV_DATE constraint , I got 2 rows as Output as below:

emp_num	First_name	LAST_name	email	tot_q
84078	DIEGO	ERWIN	E.DIEGO98@LGCOMPANY.COM	27
84106	FELICE	SAMUEL	S.FELICE98@LGCOMPANY.COM	27

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  a.Cust_Code,  a.cust_lname,  a.cust_fname  FROM  (SELECT  c.Cust_Code,
c.cust_fname, c.cust_lname FROM customer c INNER JOIN
invoice i ON c.Cust_Code = i.Customer_Cust_Code WHERE i.Employee_Emp_Num = 83649) a
INNER JOIN
(SELECT c.Cust_Code, c.cust_fname,c.cust_lname
FROM customer c INNER JOIN invoice i ON c.Cust_Code = i.Customer_Cust_Code WHERE
i.Employee_Emp_Num = 83677) b
ON a.Cust_Code=b.Cust_Code ORDER BY a.cust_lname, a.cust_fname;
```

Output:

0 rows returned.

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 cus.Cust_Code, cus.cust_fname, cus.cust_lname, cus.cust_street, cus.cust_city,
cus.cust_state, cus.cust_zip, inv1.inv_date,
MAX(COALESCE(inv1.Invoice_Total,0)) AS MAX_inv FROM customer cus left outer
JOIN invoice inv1 ON
cus.Cust_Code = inv1.Customer_Cust_Code WHERE cus.cust_state = 'AL' AND
inv1.Inv_Num not like '-%' GROUP BY cus.Cust_Code;
```

Output:

The screenshot shows a database query tool interface. The top pane displays the SQL query for Query6. The bottom pane shows the results in a grid format. The results table has columns: Cust_Code, cust_fname, cust_lname, cust_street, cust_city, cust_state, cust_zip, inv_date, and MAX_inv. The results show 50 rows of data for customers in Alabama (AL).

Cust_Code	cust_fname	cust_lname	cust_street	cust_city	cust_state	cust_zip	inv_date	MAX_inv
89	MONICA	CANTRELL	697 ADAK CIRCLE	Loachapoka	AL	36865	2014-01-12	314.25
152	LISETTE	WHITTAKER	339 NORTH PARK DRIVE	Montgomery	AL	36197	2013-11-19	139.22000122070312
169	ROSS	LANG	1991 EASTWIND COURT	Higdon	AL	35979	2013-11-14	177.30999755859375
188	LUANNE	GOODWIN	293 KIANA AVENUE	Pinegrove	AL	36507	2013-08-11	202.41000366210938
218	LUPE	SANTANA	1292 WEST 70TH PLACE	Phenix City	AL	36867	2013-10-31	270.07000732421875
219	CATHI	WHITEHEAD	760 WOODCLIFF DRIVE	Huntsville	AL	35893	2013-11-19	273.2300109863281
286	JEANNE	STEINER	1974 SCHLUSS DRIVE	Carrollton	AL	35447	2013-03-20	226.39999389648438
295	DORTHY	AUSTIN	829 BIG BEND LOOP	Diamond Shamrock	AL	36614	2013-11-01	86.54000091552734
304	GERTRUDE	CONNORS	1042 PLEASANT DRIVE	Georgiana	AL	36033	2013-12-29	376.32000732421875
364	DELLA	MAYO	543 STEPHENS CIRCLE	Birmingham	AL	35214	2013-07-08	94.93000030517578

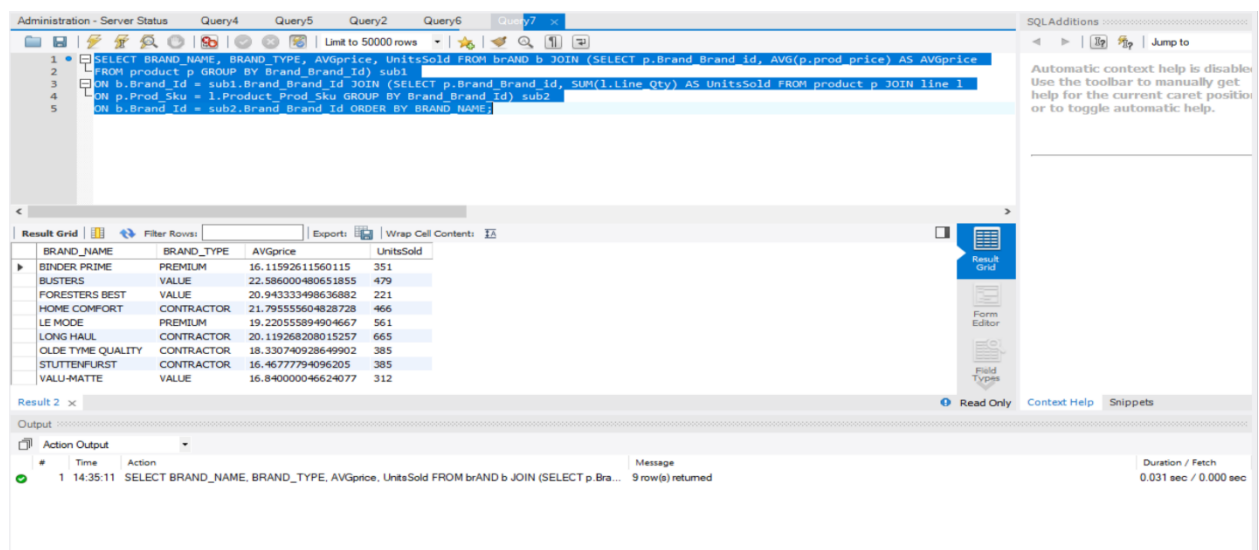
The bottom pane shows the output of the query, indicating that 50 rows were returned. The message says: "SELECT cus.Cust_Code, cus.cust_fname, cus.cust_lname, cus.cust_street, cus.cust_city, cus.cust_s... 50 row(s) returned". The duration of the query was 0.015 sec / 0.000 sec.

50 rows were returned, and output is attached in OutputQuery6.csv

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, AVGprice, UnitsSold FROM brAND b JOIN  
(SELECT p.Brand_Brand_id, AVG(p.prod_price) AS AVGprice  
FROM product p GROUP BY Brand_Brand_Id) sub1  
ON b.Brand_Id = sub1.Brand_Brand_Id JOIN (SELECT p.Brand_Brand_id, SUM(l.Line_Qty)  
AS UnitsSold FROM product p JOIN line l  
ON p.Prod_Sku = l.Product_Prod_Sku GROUP BY Brand_Brand_Id) sub2  
ON b.Brand_Id = sub2.Brand_Brand_Id ORDER BY BRAND_NAME;
```

Output:

The screenshot shows a database query tool interface. The top pane displays the SQL query, and the bottom pane shows the results in a grid format. The results table has four columns: BRAND_NAME, BRAND_TYPE, AVGprice, and UnitsSold. The data is sorted by BRAND_NAME.

BRAND_NAME	BRAND_TYPE	AVGprice	UnitsSold
BINDER PRIME	PREMIUM	16.11592611560115	351
BUSTERS	VALUE	22.586000480651855	479
FORESTERS BEST	VALUE	20.943333498636882	221
HOME COMFORT	CONTRACTOR	21.795555604828728	466
LE MOORE	PREMIUM	19.220555894904667	561
LONG HAUL	CONTRACTOR	20.119268208015257	665
OLDE TIME QUALITY	CONTRACTOR	18.330740928649902	385
STUTTENFURST	CONTRACTOR	16.46777794096205	385
VALU-MATTE	VALUE	16.840000046624077	312

Result 2 x

Output

1 14:35:11 SELECT BRAND_NAME, BRAND_TYPE, AVGprice, UnitsSold FROM brAND b JOIN (SELECT p.Br... 9 row(s) returned

Duration / Fetch 0.031 sec / 0.000 sec

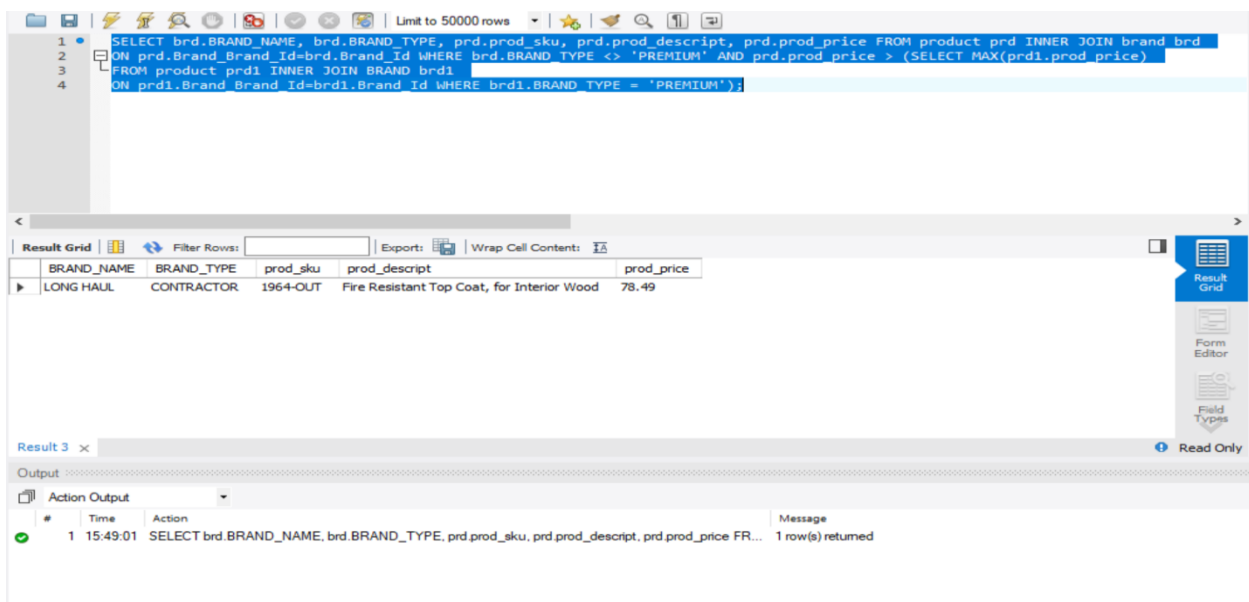
9 rows were returned as attached.

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 brd.BRAND_NAME, brd.BRAND_TYPE, prd.prod_sku, prd.prod_descript,  
prd.prod_price FROM product prd INNER JOIN brand brd  
ON prd.Brand_Brand_Id=brd.Brand_Id WHERE brd.BRAND_TYPE <> 'PREMIUM'  
AND prd.prod_price > (SELECT MAX(prd1.prod_price)  
FROM product prd1 INNER JOIN BRAND brd1  
ON prd1.Brand_Brand_Id=brd1.Brand_Id  
WHERE brd1.BRAND_TYPE = 'PREMIUM');
```

Ouput:



The screenshot displays a database query tool interface. The top section shows the SQL query being executed. Below the query, the 'Result Grid' is visible, showing a single row of data. The 'Output' section at the bottom shows the execution details, including the time taken and the number of rows returned.

BRAND_NAME	BRAND_TYPE	prod_sku	prod_descript	prod_price
LONG HAUL	CONTRACTOR	1964-OUT	Fire Resistant Top Coat, for Interior Wood	78.49

Result 3 x

Output

Action Output

#	Time	Action	Message
1	15:49:01	SELECT brd.BRAND_NAME, brd.BRAND_TYPE, prd.prod_sku, prd.prod_descript, prd.prod_price FR...	1 row(s) returned

There is only 1 such 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

BRAND_NAME	BRAND_TYPE	prod_sku	prod_d escript	prod_price
LONG HAUL	CONTRACTOR	1964-OUT	Fire Resistan t Top Coat, for Interior Wood	78.49

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 WHERE prod_price > 50;

Output:

The screenshot shows a SQL query execution interface. The query is: `SELECT * FROM product WHERE prod_price > 50;`. The results are displayed in a table with columns: Prod_Sku, Prod_Descript, Prod_Type, Prod_Base, Prod_Category, Prod_Price, Prod_QOH, Prod_Min, and Brand_Brand_Id. The results show three products with prices greater than \$50: 1021-MTI (Elastic, Exterior, Industrial Grade, Water B...), 1964-OUT (Fire Resistant Top Coat, for Interior Wood), and 3694-XFJ (Epoxy-Modified Latex, Interior, Semi-Gloss (MPI...)).

Prod_Sku	Prod_Descript	Prod_Type	Prod_Base	Prod_Category	Prod_Price	Prod_QOH	Prod_Min	Brand_Brand_Id
1021-MTI	Elastic, Exterior, Industrial Grade, Water B...	Exterior	Water	Top Coat	62.99	22	25	35
1964-OUT	Fire Resistant Top Coat, for Interior Wood	Interior	Solvent	Top Coat	78.49	120	10	30
3694-XFJ	Epoxy-Modified Latex, Interior, Semi-Gloss (MPI...	Interior	Water	Top Coat	54.89	39	25	27

The interface also shows a log of the query execution, including the time taken and the number of rows returned.

#	Time	Action	Message	Duration / Fetch
1	15:49:01	SELECT brd.BRAND_NAME, brd.BRAND_TYPE, prd.prod_sku, prd.prod_desc, prd.prod_price FR...	1 row(s) returned	0.000 sec / 0.000 sec
2	15:54:45	SELECT * FROM product WHERE prod_price > 50 LIMIT 0, 50000	3 row(s) returned	0.000 sec / 0.000 sec
3	15:55:14	SELECT * FROM product WHERE prod_price > 50 LIMIT 0, 50000	3 row(s) returned	0.000 sec / 0.000 sec

There are 3 products with price greater than \$50 as below

Prod_Sku	Prod_Descript	Prod_Type	Prod_Base	Prod_Category	Prod_Price
1021-MTI	Elastomeric, Exterior, Industrial Grade, Water Based	Exterior	Water	Top Coat	62.99
1964-OUT	Fire Resistant Top Coat, for Interior Wood	Interior	Solvent	Top Coat	78.49
3694-XFJ	Epoxy-Modified Latex, Interior, Semi-Gloss (MPI Gloss Level 5)	Interior	Water	Top Coat	54.89

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

Query:

SELECT SUM(prod_qoh*prod_price) AS 'Total_Value' FROM product;

Output:

The screenshot displays a SQL query execution environment. The query editor shows the following SQL code:

```

1  # (i)
2  SELECT * FROM product WHERE prod_price > 50;
3  # (ii)
4  SELECT SUM(prod_qoh*prod_price) AS 'Total_Value' FROM product;
5  # (iii)
6  SELECT count(distinct(c.Cust_Code)) AS "Count_Customers", SUM(c.cust_balance) AS "Total_Cus_Bal" FROM customer c;
7  # (iv)
8  SELECT cus.CUST_STATE,SUM(inv.INV_TOTAL) AS inv_tot FROM customer cus INNER JOIN invoice inv
9  ON cus.CUST_CODE=inv.CUST_CODE WHERE inv.Invoice_Inv_Num not like '-%'
10 GROUP BY cus.CUST_STATE
11 ORDER BY inv_tot desc limit 3;

```

The results pane shows a single row for the total value:

Total_Value
360307.7926878929

The output pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	15:49:01	SELECT brd.BRAND_NAME, brd.BRAND_TYPE, prd.prod_sku, prd.prod_descript, prd.prod_price FR...	1 row(s) returned	0.000 sec / 0.000 sec
2	15:54:45	SELECT * FROM product WHERE prod_price > 50 LIMIT 0, 50000	3 row(s) returned	0.000 sec / 0.000 sec
3	15:55:14	SELECT * FROM product WHERE prod_price > 50 LIMIT 0, 50000	3 row(s) returned	0.000 sec / 0.000 sec
4	16:00:46	SELECT SUM(prod_qoh*prod_price) AS 'Total_Value' FROM product LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec

The total value of the inventory is : \$360307.7926878929

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

Query:

```
SELECT count(distinct(c.Cust_Code)) AS "NumberofCustomers", SUM(c.cust_balance)
AS "TotalofAll_Customer_Balances"
FROM customer c;
```

Output:

The screenshot shows a database management interface with a SQL query editor and a results pane. The query is as follows:

```
1 # (i)
2 SELECT * FROM product WHERE prod_price > 50;
3 # (ii)
4 SELECT SUM(prod_goh*prod_price) AS 'Total_Value' FROM product;
5 # (iii)
6 SELECT count(distinct(c.Cust_Code)) AS "NumberofCustomers", SUM(c.cust_balance) AS "TotalofAll_Customer_Balances" FROM customer c
7 # (iv)
8 SELECT cus.CUST_STATE,SUM(inv.INV_TOTAL) AS inv_tot FROM customer cus INNER JOIN invoice inv
9 ON cus.CUST_CODE=inv.CUSTOMER_CODE WHERE inv.Invoice_Inv_Num not like '-%'
10 GROUP BY cus.CUST_STATE
11 ORDER BY inv_tot desc limit 3;
```

The results pane displays the following data:

NumberofCustomers	TotalofAll_Customer_Balances
1362	787201.150834322

The bottom pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	15:49:01	SELECT brd.BRAND_NAME, brd.BRAND_TYPE, prd.prod_sku, prd.prod_descript, prd.prod_price FR...	1 row(s) returned	0.000 sec / 0.000 sec
2	15:54:45	SELECT * FROM product WHERE prod_price > 50 LIMIT 0, 50000	3 row(s) returned	0.000 sec / 0.000 sec
3	15:55:14	SELECT * FROM product WHERE prod_price > 50 LIMIT 0, 50000	3 row(s) returned	0.000 sec / 0.000 sec
4	16:00:46	SELECT SUM(prod_goh*prod_price) AS 'Total_Value' FROM product LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec
5	16:03:33	SELECT count(distinct(c.Cust_Code)) AS "NumberofCustomers", SUM(c.cust_balance) AS "TotalofAll...	1 row(s) returned	0.000 sec / 0.000 sec

There are a total of 1362 customers and the

Total of all Customer Balances is \$787201.150834322

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

Query:

```
SELECT cus.CUST_STATE,SUM(inv.Invoice_Total) AS inv_tot FROM customer cus
INNER JOIN invoice inv
ON cus.Cust_Code=inv.Customer_Cust_Code WHERE inv.Inv_Num not like '-%'
GROUP BY cus.CUST_STATE
ORDER BY inv_tot desc limit 3;
```


Output:

The screenshot shows a database query editor with a query window titled 'Query9'. The SQL code is as follows:

```

1  # (i)
2  SELECT * FROM product WHERE prod_price > 50;
3  # (ii)
4  SELECT SUM(prod_qoh*prod_price) AS 'Total_Value' FROM product;
5  # (iii)
6  SELECT count(distinct(c.Cust_Code)) AS "NumberofCustomers", SUM(c.cust_balance) AS "TotalofAll_Customer_Balances" FROM customer c
7  # (iv)
8  SELECT cus.CUST_STATE,SUM(inv.Invoice_Total) AS inv_tot FROM customer cus INNER JOIN invoice inv
9  ON cus.Cust_Code=inv.Customer Cust Code WHERE inv.Inv_Num not like '-%'
10 GROUP BY cus.CUST_STATE
11 ORDER BY inv_tot desc limit 3;

```

The results are displayed in a table with two columns: CUST_STATE and inv_tot. The top three states are PA, NY, and NC.

CUST_STATE	inv_tot
PA	38618.11987400055
NY	32242.929913520813
NC	19611.390100479126

The bottom of the screenshot shows the 'Output' section with a message: '1 16:25:41 SELECT cus.CUST_STATE,SUM(inv.Invoice_Total) AS inv_tot FROM customer cus INNER JOIN inv... 3 row(s) returned'.

The top 3 states that buy the most product in dollars from the company are PA,NY and NC with the invoice totals as below:

CUST_STATE	inv_tot
PA	38618.12
NY	32242.93
NC	19611.39

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.)

```
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 R^2 , model, coefficients, and the confidence interval for your analysis.

Ans:

I converted the DATETIME values to Julian Date Time and then used R to evaluate various models.

The R Code gave very bad results as our data is time series data.

Hence, I converted the whole data into months and used the same for prediction. Please find the model summary and Results below

SUMMARY OUTPUT REGRESSION

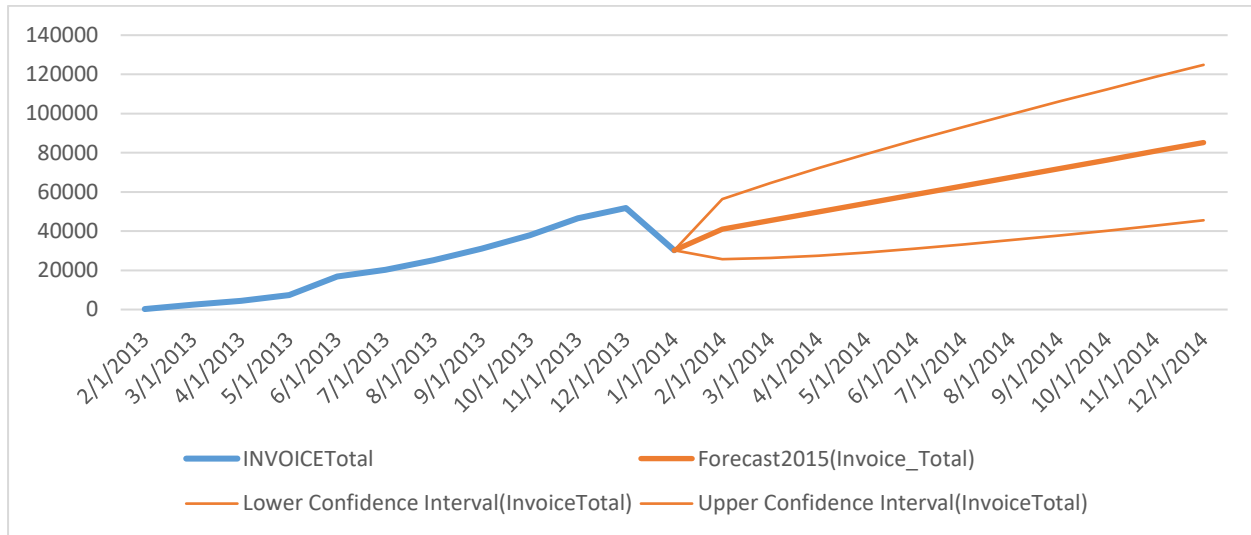
<i>Regression Statistics</i>	
Multiple R	0.03802
R Square	0.00145
Adjusted R Square	0.0007
Standard Error	128.892
Observations	1346

ANOVA					
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>
Regression	1	32322.66263	32322.66263	1.945597976	0.163292475
Residual	1344	22328178.34	16613.22793		
Total	1345	22360501			

	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>
Intercept	-2567.4	1986.876061	-1.292174585	0.196518821	-6465.106377
41317	0.06669	0.047813537	1.394846936	0.163292475	-0.027104715

2015 Forecasting Invoice Total:

The given data is a time-series , hence I used excel to use time-series to forecast of next year sales based on the monthly sum total of sales and dividing into monthly chunks.

**MODEL SUMMARY FOR R**

From the above table, We see that linear model has the highest R^2 and hence is the best model. We have the least AIC,BIC values from which we select the best model with the least values. Here the linear model has the AIC BIC values with 17099.97,17115.62.

Model Name	R^2	Alkaline Information Criteria(AIC)	Bayesian Information Criteria(BIC)
Model_Linear	0.01357	17099.97	17115.62
Model_Quadratic	0.01356	17099.98	17115.63
Model_Cubic	0.0006201	17099.98	17115.63
Model_Differencing Time Series)	4.877e-05	17995	18010.65

SUMMARY OF MODEL LINEAR

```
Console Terminal R Markdown
C:/Data Analytics/FALL2018UTD/BAwithR_LingGe/Assignments/HW3/

> summary(Model_Linear)

Call:
lm(formula = inv_total ~ inv_date, data = Invoice)

Residuals:
    Min       1Q   Median       3Q      Max
-203.46 -102.57  -13.67   77.55  488.89

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.475e+03  1.970e+03  -1.256   0.209
inv_date      6.445e-02  4.741e-02   1.360   0.174

Residual standard error: 128.7 on 1360 degrees of freedom
Multiple R-squared:  0.001357, Adjusted R-squared:  0.0006231
F-statistic: 1.849 on 1 and 1360 DF, p-value: 0.1742

> AIC(Model_Linear)
[1] 17099.97
> BIC(Model_Linear)
[1] 17115.62
> |
```

SUMMARY OF MODEL QUADRATIC

```
C:/Data Analytics/FALL2018UTD/BAwithR_LingGe/Assignments/HW3/

> summary(Model_Quadratic)

Call:
lm(formula = inv_total ~ I(inv_date^2), data = Invoice)

Residuals:
    Min       1Q   Median       3Q      Max
-203.46 -102.58  -13.66   77.54  488.90

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.136e+03  9.856e+02  -1.153   0.249
I(inv_date^2)  7.756e-07  5.707e-07   1.359   0.174

Residual standard error: 128.7 on 1360 degrees of freedom
Multiple R-squared:  0.001356, Adjusted R-squared:  0.0006216
F-statistic: 1.846 on 1 and 1360 DF, p-value: 0.1744

> AIC(Model_Quadratic)
[1] 17099.98
> BIC(Model_Quadratic)
[1] 17115.63
> |
```

SUMMARY OF MODEL_CUBIC

```
C:/Data Analytics/FALL2018UTD/BAwithR_LingGe/Assignments/HW3/
> summary(Model_Cubic)

Call:
lm(formula = inv_total ~ I(inv_date^3), data = Invoice)

Residuals:
    Min       1Q   Median       3Q      Max
-203.46 -102.59  -13.66   77.54  488.90

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -6.897e+02  6.574e+02  -1.049   0.294
I(inv_date^3)  1.244e-11  9.162e-12   1.358   0.175

Residual standard error: 128.7 on 1360 degrees of freedom
Multiple R-squared:  0.001354, Adjusted R-squared:  0.0006201
F-statistic: 1.844 on 1 and 1360 DF,  p-value: 0.1747

> AIC(Model_Cubic)
[1] 17099.98
> BIC(Model_Cubic)
[1] 17115.63
~
```

SUMMARY OF MODEL_DIFFERENCING

```
C:/Data Analytics/FALL2018UTD/BAwithR_LingGe/Assignments/HW3/
> summary(Model_Differencing)

Call:
lm(formula = diff(inv_total) ~ diff(inv_date), data = Invoice)

Residuals:
    Min       1Q   Median       3Q      Max
-616.06 -117.79    0.88  120.52  544.83

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.006656  4.867366  -0.001   0.999
diff(inv_date) -0.015095  0.058632  -0.257   0.797

Residual standard error: 179.6 on 1359 degrees of freedom
Multiple R-squared:  4.877e-05, Adjusted R-squared: -0.000687
F-statistic: 0.06628 on 1 and 1359 DF,  p-value: 0.7969

> AIC(Model_Differencing)
[1] 17995
> BIC(Model_Differencing)
[1] 18010.65
~
```

R Code for Prediction:

Question10DB

Phanindra

December 7, 2018

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(data.table)
library(DBI)
library(RSQLite)
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.2.1 --

## v ggplot2 3.1.0      v purrr  0.2.5
## v tibble  1.4.2      v dplyr  0.7.8
## v tidyr   0.8.2      v stringr 1.3.1
## v readr   1.1.1      v forcats 0.3.0

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()

library(ggplot2)
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##   as.Date, as.Date.numeric
```

```
library(sandwich)
library(margins)
library(partykit)

## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm

library(plm)

## Loading required package: Formula

##
## Attaching package: 'plm'

## The following objects are masked from 'package:dplyr':
##
##   between, lag, lead

## The following object is masked from 'package:data.table':
##
##   between

library(dplyr)

Invoice <- fread("Invoice.csv")
View(Invoice)
inv_date <- Invoice$`JULIAN INV_DATE`
class(inv_date)

## [1] "integer"

inv_date <- as.numeric( Invoice$`JULIAN INV_DATE`)
class(inv_date)

## [1] "numeric"

inv_total <- Invoice$INV_TOTAL
class(inv_total)

## [1] "numeric"

Model_Linear<- lm(inv_total~ inv_date, data=Invoice)
summary(Model_Linear)#Multiple R-squared:  0.001357,    Adjusted R-squared:
0.0006231

##
## Call:
## lm(formula = inv_total ~ inv_date, data = Invoice)
##
## Residuals:
```

```
##      Min      1Q  Median      3Q      Max
## -203.46 -102.57  -13.67   77.55  488.89
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.475e+03  1.970e+03  -1.256    0.209
## inv_date      6.445e-02  4.741e-02   1.360    0.174
##
## Residual standard error: 128.7 on 1360 degrees of freedom
## Multiple R-squared:  0.001357,    Adjusted R-squared:  0.0006231
## F-statistic: 1.849 on 1 and 1360 DF,  p-value: 0.1742

AIC(Model_Linear)#17099.97

## [1] 17099.97

BIC(Model_Linear)#17115.62

## [1] 17115.62

Model_Quadratic <- lm(inv_total~ I(inv_date^2), data=Invoice)
summary(Model_Quadratic)#Multiple R-squared:  0.001356, Adjusted R-squared:
0.0006216

##
## Call:
## lm(formula = inv_total ~ I(inv_date^2), data = Invoice)
##
## Residuals:
##      Min      1Q  Median      3Q      Max
## -203.46 -102.58  -13.66   77.54  488.90
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.136e+03  9.856e+02  -1.153    0.249
## I(inv_date^2)  7.756e-07  5.707e-07   1.359    0.174
##
## Residual standard error: 128.7 on 1360 degrees of freedom
## Multiple R-squared:  0.001356,    Adjusted R-squared:  0.0006216
## F-statistic: 1.846 on 1 and 1360 DF,  p-value: 0.1744

AIC(Model_Quadratic)#17099.98

## [1] 17099.98

BIC(Model_Quadratic)#17115.63

## [1] 17115.63

Model_Cubic <- lm(inv_total~ I(inv_date^3), data=Invoice)
summary(Model_Cubic)#Multiple R-squared:  0.001354, Adjusted R-squared:  0.00
06201
```



```
##
## Call:
## lm(formula = inv_total ~ I(inv_date^3), data = Invoice)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -203.46 -102.59  -13.66   77.54  488.90
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.897e+02  6.574e+02  -1.049   0.294
## I(inv_date^3)  1.244e-11  9.162e-12   1.358   0.175
##
## Residual standard error: 128.7 on 1360 degrees of freedom
## Multiple R-squared:  0.001354, Adjusted R-squared:  0.0006201
## F-statistic: 1.844 on 1 and 1360 DF, p-value: 0.1747

AIC(Model_Cubic)#17099.98

## [1] 17099.98

BIC(Model_Cubic)#17115.63

## [1] 17115.63

Model_Differencing <- lm(diff(inv_total)~ diff(inv_date), data=Invoice)
summary(Model_Differencing)#Multiple R-squared:  4.877e-05, Adjusted R-square
d:  -0.000687

##
## Call:
## lm(formula = diff(inv_total) ~ diff(inv_date), data = Invoice)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -616.06 -117.79    0.88  120.52  544.83
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.006656   4.867366  -0.001   0.999
## diff(inv_date) -0.015095   0.058632  -0.257   0.797
##
## Residual standard error: 179.6 on 1359 degrees of freedom
## Multiple R-squared:  4.877e-05, Adjusted R-squared:  -0.000687
## F-statistic: 0.06628 on 1 and 1359 DF, p-value: 0.7969

AIC(Model_Differencing)#17995

## [1] 17995

BIC(Model_Differencing)#18010.65

## [1] 18010.65
```

The best model based on prediction came out to be Linear model. Using it for predicting the values for 2015.

```
#The Julian date for 2/12/2013 based on the previous invoice is 41317. Julian date is the number of days since 1/1/1990.
#So for 1/1/2015 Julian Date=41317+(Days difference between 2/12/2013 and 1/1/1990)
JulianDate2015firstday <- 41317+(365-(28+31))+365 #41988
JulianDate2015lastday <- JulianDate2015firstday+365
JulianVector2015 <- seq(from=JulianDate2015firstday,to=JulianDate2015lastday,by=1)

#Predicting values for 2015
InvoiceTotal2015 <- predict(Model_Linear,data=JulianVector2015)

PredictedValuesfor2015=list("JulianDate2015"=JulianVector2015,"Predicted2015INVTOTAL"=InvoiceTotal2015)
```

SPLITTING INTO DIFFERENT TABLES R CODE :

RDatabaseBUAN6320

Phanindra

November 28, 2018

```
dataset2<- read.delim("BUAN6320-DataSet2.txt",header=TRUE,sep="\t",
  dec = ".", fill = TRUE, comment.char = "")
dataset3<-read.delim("BUAN6320-DataSet3.txt",header=TRUE,sep="\t",
  dec = ".", fill = TRUE, comment.char = "")
dataset4<-read.delim("BUAN6320-DataSet4.txt",header=TRUE,sep="\t",
  dec = ".", fill = TRUE, comment.char = "")
dataset2omitted <- na.omit(dataset2)
dataset3omitted <- na.omit(dataset3)
dataset4omitted <- na.omit(dataset4)
#Dataset3 has columns X,X.1,X.2 which have all NA values, SO they have to be removed.
dataset3removedX <- dataset3[,-c(18,19,20)]
dataset3omitted <- na.omit(dataset3removedX)
```

```
#Now dataset3 has all the 252 observations which were present initially.
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union

vendor <- dataset3omitted %>%
  dplyr::select(VEND_STREET,VEND_NAME,VEND_STATE,VEND_CITY,VEND_ID,VEND_ZIP)
#Now exporting the vendor file as a CSV file
write.csv(vendor,file="vendor.csv")

#Now finding the customer table as designed
customer <- dataset2omitted %>%
  dplyr::select(CUST_CODE,CUST_FNAME,CUST_LNAME,CUST_STREET,CUST_CITY,CUST_ST
ATE,CUST_ZIP,CUST_BALANCE)
write.csv(customer,file="customer.csv")

#Getting the data for brand
brand <- dataset3omitted %>%
  dplyr::select(BRAND_NAME,BRAND_TYPE,BRAND_ID)
write.csv(brand,file="brand.csv")
#Getting data for product
product <- dataset3omitted %>%
  dplyr::select(PROD_SKU,PROD_DESCRIPT,PROD_TYPE,PROD_BASE,PROD_CATEGORY,PROD
_PRICE,PROD_QOH,PROD_MIN,BRAND_ID)
write.csv(product,file="product.csv")

##Getting invoice
invoice <- dataset2omitted %>%
  dplyr::select(INV_NUM,INV_DATE,INV_TOTAL,EMPLOYEE_ID,CUST_CODE)
write.csv(invoice,file="invoice.csv")

##After getting all the csv files, duplicates were removed in excel using rem
ove duplicates in Data tab.
#Getting data for department
Department <- dataset4omitted %>%
  dplyr :: select ( DEPT_NUM, DEPT_NAME, DEPT_MAIL_BOX, DEPT_PHONE, SUPV_EMP_
NUM)
write.csv(Department, file= "Department.csv")
```

```
##Getting salary history
Salary_history <- dataset4omitted %>%
  dplyr :: select (SAL_FROM,SAL_END,SAL_AMOUNT,EMP_NUM)
write.csv(Salary_history, file= "Salary_history.csv")

##Getting line
line <- dataset2omitted %>%
  dplyr :: select (LINE_NUM,LINE_QTY,LINE_PRICE,INV_NUM,PROD_SKU)
write.csv(line, file= "line.csv")

##Supplies
Supplies <- dataset3omitted %>%
  dplyr :: select (PROD_SKU,VEND_ID)
write.csv(Supplies,file="supplies.csv")
```

BUAN 6320

Individual Project

Due: 12/7/18