University of California

Berkeley

Uptown Juice Company Database Prototype Design

Fall 2016

IEOR 115 Industrial and Commercial Data Systems

Industrial Engineering and Operations Research Department

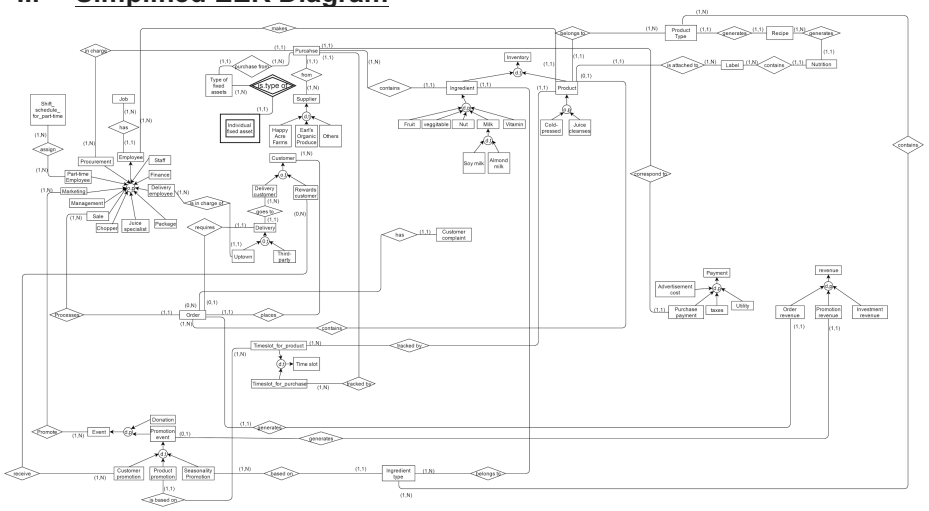
By

Guo Chen, Xuewei Du, Dongyu Lang, Soyoung Lee, Anna Liu, Javier Sanchez, Huidi Wang, Weiwen-Abby Zhang

I. Client Description

Uptown Juice Company was founded in 2013. It is a local business that produces nutrient and organic cold-pressed juices for people who are seeking a healthy regimen. It is committed to work with local farms whenever possible in order to ensure an environment-friendly business approach. There is one retail store located in 1629 Broadway in Oakland and it offers delivery services through Caviar and UberEats and its website is available at http://www.uptownjuice.com/.

II. Simplified EER Diagram

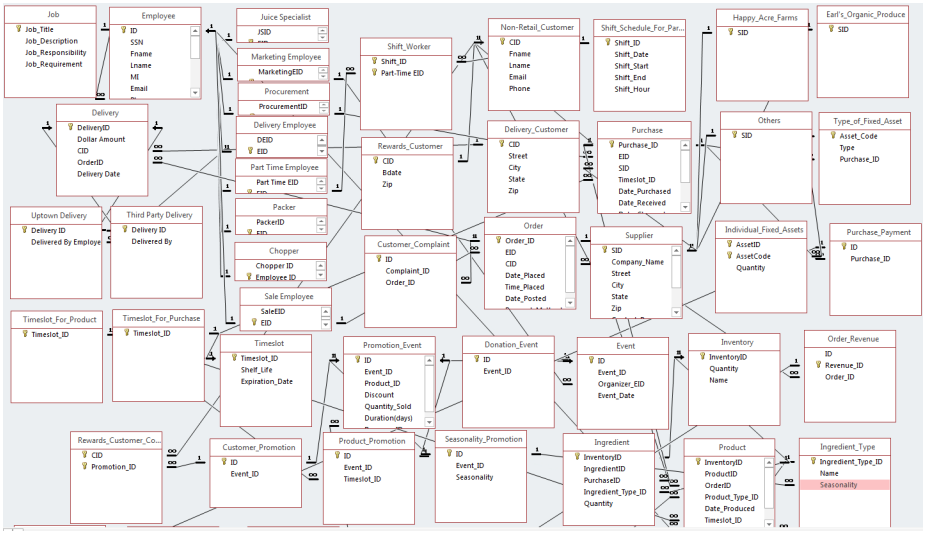


III. Original Relational Design (Updated Normalized Relation Design is attached in the appendix)

1. Employee (EID, Fname, Lname, MI, Emaill, Phone, SSN, Street, City, State, Zip, Bdate, Job_Title[2], Salary)
    - 1a. Chopper (ChopperID,Speed, EID[1a])
    - 1b. Juice_Specialist (JSID, rating, experience, EID[1a])
    - 1c. Packer (PackerID,Efficiency, EID[1a])
    - 1d Part_Time_Employee (PTEID, EID[1a], Hourly_Wage, Availabilities,)
    - 1e Marketing_Employee(MarketingID,Flyer_Quantity, EID[1a])
    - 1f Procurement(ProcurementID,Business_Card_Quantity, EID[1a])
    - 1g Delivery_Employee(DEID, EID[1a], DriversLicense)
    - 1h Sale_Employee(SaleEID, EID[1a], Commission)
2. Job (Job_Title, Job_Description, Job_Responsibilities, Job_Requirements)
3. Non-Retail_Customer (CID, Fname, Lname, Email, Phone)
    - 3a. Delivery_Customer(CID[3], Street, City, State, Zip)
    - 3b. Rewards_Customer (CID[3], Bdate)
4. Rewards_Customer_Coupon(CID[3], Promotion_ID[24a])
5. Delivery (Delivery_ID, Dollar_Amount, CID[3], Order_ID[19], Delivery_Date)
    - 5a. UPtown_Delivery (Delivery_ID[5], Delivered_By_Employee[1a])
    - 5b. Third_Party_Delivery (Delivery_ID[5], Delivered_By) (dom(Delivered_By) ={Caviar, UberEats})

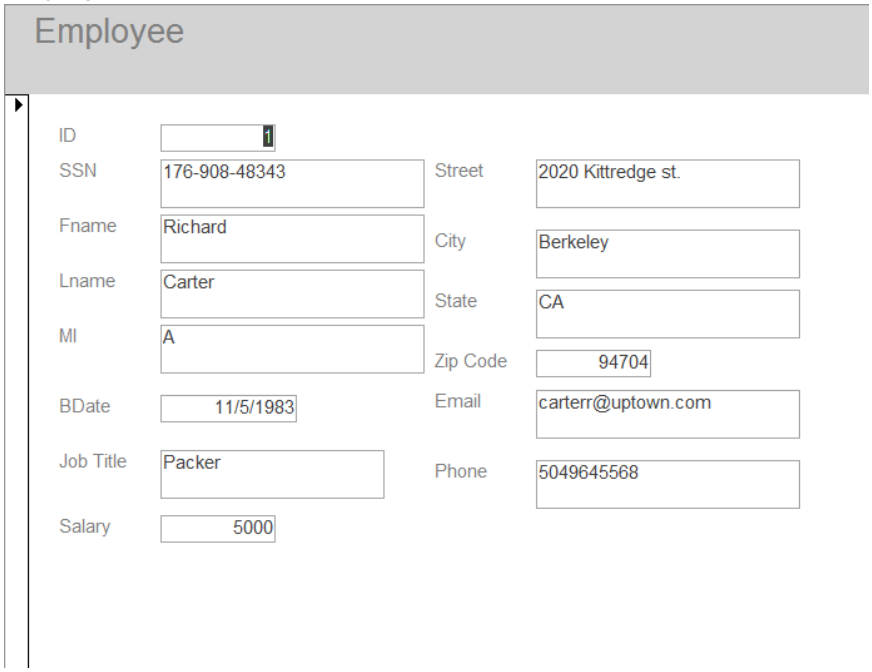
6. Recipe(RecipeID, Ingredient_ID[12a], Quantity)
7. Label (Label_ID,Attachment1, Attachment2)
8. Nutrition_Fact (Nutrition_ID, Nutrition_Name, Nutrition_Value)
9. Supplier (SID, Company_Name, Street, City, State, Zip, Contact_Person, Phone, Email, Discount)
10. Type_of_Fixed_Asset(Asset_Code (1 to 12),Type, Purchase_ID)
dom(Type)= {Office, Kitchen, Vehicle, }
11. Individual_Fixed_Asset(Asse_ID, Asset_Code[10], Quantity)
12. Inventory (Inventory_ID, Quantity, Name)
12a. Ingredient (Inventory_ID[12], Ingredient_ID, Purchase_ID[15], Ingredient_Type_ID[13], Quantity)
12b. Product (Inventory_ID[12], Product_ID, Order_ID[20], Product_Type_ID[14],Date_Produced, Timeslot_ID[17], EID[1a])
13. Ingredient_Type(Ingredient_Type_ID, Name, Seasonality)
dom(Seasonality)={spring,summer,fall,winter}
14. Product_Type(Product_Type_ID, Product_Name, RecipeID)
15. Purchase (Purchase_ID, EID[1a], SID[9], Timeslot_ID[17], Date_Purchased, Date_Received, Date_Charged, Quantity, Type)
(dom(Type)={Ingredient, FixedAsset})
16. Product_Detail (Ingredient_ID, Product_ID)
17. Timeslot (Timeslot_ID, Expiration_Date)
17a. Timeslot_For_Purchase (Timeslot_ID[17])
17b. Timeslot_For_Product (Timeslot_ID[17],Shelf_Life)
18. Shift_Schedule_For_Part_Time_Employee (Shift_ID, Date, Shift_Start, Shift_End, Hour)
19. Shift_Worker(Shift_ID, EID[1a])
20. Order (Order_ID, EID[1a], CID[3], Date_Placed, Time_Placed, Date_Posted, Payment_Method, Day, Time) (dom(Time)={Morning, Afternoon, Night}, dom(Day)={Monday,Tuesday,Wednesday,Thursday,Friday}
21. Order_Details (Order_ID[20], Product_ID[12b], Quantity, UnitPrice)
22. Payment(Payment_ID, DollarAmount)
22a. Purchase_Payment (Payment_ID,Purchase_ID[15])
22b. Utilities_Payment(Payment_ID, Utility_Type)
23. Revenue(Revenue_ID, DollarAmount)
23a. Order_Revenue (Revenue_ID, Order_ID[20])
23b. Promotion_Revenue (Revenue_ID, Event_ID[24])
24. Event (Event_ID, Organizer_EID[2](EID), Event_Date)
24a) Promotion_Event(Event_ID[24], Product_ID[12b], Discount, Quantity_Sold, Duration, Revenue_ID)
24aa. Customer_Promotion (Event_ID[24])
24ab. Product Promotion (Event_ID[24], Timeslot_ID[17b])
24ac. Seasonality_Promotion (Event_ID[24],Seasonality[13]
24b) Donation_Event (Event_ID[24])
25. Customer_Complaint (Complaint_ID, Order_ID[20])

IV. Relational Design Implemented in Access



V. Forms and Reports

1. Forms



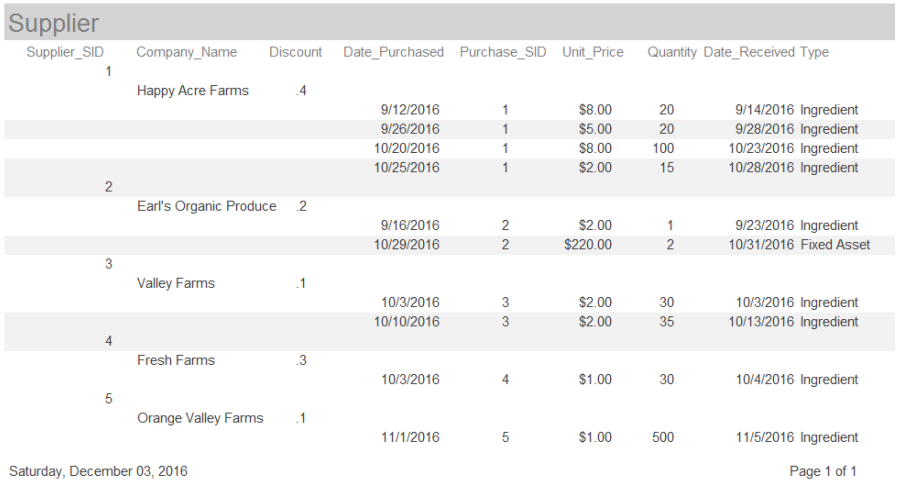
Supplier

SID	1
Company_Name	Happy Acre Farms
Street	1414 Fulton St.
City	Modesto
State	California
Zip	94703
Contact_Person	James Smith
Phone	5103262551
Email	j.smith@happyacrefarms.com
Discount	.4

2. Reports

Ingredient

Timeslot_ID	Expire	Date_Purchased	Unit_Price	Ingredient_ID	Purchase_ID	Quantity	Name
3	9/30/2016						
		9/26/2016	$5.00				
				3			
					7	11	pineapple
					3	3	pineapple
5	10/18/2016						
		10/3/2016	$2.00				
				4			
					4	16	coconut
				5			
					8	15	guava
6	10/27/2016						
		10/3/2016	$1.00				
				1			
					9	13	apple
					5	34	apple
7	9/22/2016						
		9/12/2016	$8.00				
				1			
					10	14	apple
					1	0	apple
8	9/18/2016						
		9/16/2016	$2.00				
				2			
					6	23	orange
					2	2	orange



Supplier

Supplier_SID	Company_Name	Discount	Date_Purchased	Purchase_SID	Unit_Price	Quantity	Date_Received	Type
1								
	Happy Acre Farms	.4						
			9/12/2016	1	$8.00	20	9/14/2016	Ingredient
			9/26/2016	1	$5.00	20	9/28/2016	Ingredient
			10/20/2016	1	$8.00	100	10/23/2016	Ingredient
			10/25/2016	1	$2.00	15	10/28/2016	Ingredient
2								
	Earl's Organic Produce	.2						
			9/16/2016	2	$2.00	1	9/23/2016	Ingredient
			10/29/2016	2	$220.00	2	10/31/2016	Fixed Asset
3								
	Valley Farms	.1						
			10/3/2016	3	$2.00	30	10/3/2016	Ingredient
			10/10/2016	3	$2.00	35	10/13/2016	Ingredient
4								
	Fresh Farms	.3						
			10/3/2016	4	$1.00	30	10/4/2016	Ingredient
5								
	Orange Valley Farms	.1						
			11/1/2016	5	$1.00	500	11/5/2016	Ingredient

Saturday, December 03, 2016 Page 1 of 1

The two forms are employees' details and suppliers' details. The first report is the purchases of each ingredient with the quantities, type, date purchased and date expired. The second report is the purchases from each supplier with the company name, data purchase, unit price, quantity and data received.

VI. Queries

1. Geographical Demand Analysis

This query extracts zip code data from the rewards customer information in the database and combines it with spending information. The objective of this query is to determine where Uptown Juice Company's customers are ordering from and how much they are spending. By plotting this data on a map and analyzing geographical demand, we aim to have a better understanding of how demand for our client's products is distributed. Also, by using location planning models (demand weighted P-Median and P-Center) we plan to offer the client advice on where to optimally locate additional stores in case they choose to expand. Having this information available is of tremendous use for our client since they will be able to decide where to open up additional locations.

```
SQL  >SELECT DC.zip, sum(R.DollarAmount)
     FROM Order_Revenue as RR, Revenue as R, Order1 as O, Delivery_Customer as DC,
Delivery as D
     WHERE R.Revenue_ID = RR.Revenue_ID and OR.Order_ID = O.Order_ID and D.CID =
DC.CID and O.Order_ID = D.Order_ID
     GROUP BY DC.zip;
```

2. Contaminated Product

This query extracts each contaminated product, then gets the contaminated batch of same product type that was made on the same day. By keeping track of the ingredients being used, we can trace back to suppliers where we purchased these ingredients from. This will allow

us to detect any possible trend from these suppliers. Also, our ultimate goal is to contact customers who have purchased products from the contaminated batch to minimize the damage.

- The following SQL allows us to find the number of purchases we made from suppliers who supplied us the ingredients that were used in the contaminated batch. Having this information on hand, we can easily plot the count numbers and see if there is any trend from the supplier.

```
SQL > SELECT Purchase.SID, sum(Purchase.Quantity) as Purchase_num
      FROM Customer_Complaint CC, Product1 P1, Product1 P2, Product_Detail PD,
           Ingredient as IG, Purchase
      WHERE CC.Order_ID = P1.Order_ID and P1.Product_Type_ID = P2.Product_Type_ID
            and P1.Date_Produced = P2.Date_Produced
            and P2.Product_ID = PD.Product_ID and PD.Ingredient_ID = IG.Ingredient_ID
            and IG.Purchase_ID = Purchase.Purchase_ID
      GROUP BY P2.Date_Produced, P2.Product_Type_ID, Purchase.SID;
```

- The following SQL allows us to find the information of customers who purchased products that are from the contaminated batch.

```
SQL > SELECT  NRC.Fname, NRC.Lname, NRC.Email, NRC.Phone
      FROM Non_Retail_Customer NRC, Customer_Complaint CC, Order1 O, Product1 P1,
Product1 P2
      WHERE CC.Order_ID = P1.Order_ID and P1.Product_type_ID = P2.Product_Type_ID
            and P1.Date_Produced = P2.Date_Produced and P2.Order_ID = O.Order_ID
            and O.CID = NRC.CID;
```

3. Marketing Analysis- Product Promotion

This query extracts product and discount information from product promotion event table, along with the order detail table, product table, product detail table, recipe table, ingredient table and order table. By implementing the query in SQL and analysis in R, we are able to determine product promotion effectiveness in terms of maximizing the promotion profit of the fresh juice due to its characteristic of perishability for our client. The following SQL allows us to extract necessary information in year 2016 for data analysis and modeling in next section.

```
SELECT E.Event_ID, Pr.Product_ID, Pr.Product_Type_ID,
       PE.Duration, PE.Discount, OD.UnitPrice,
       (P.Unit_Price * R.Quantity) as ing_cost
FROM Purchase as P, Product_Type as PT,
     Promotion_Event as PE, Product_Promotion as PP,
     Order_Details as OD, Product1 as Pr,
     Product_Detail as PD, Recipe as R,
     Ingredient as Ing, Order1 as O, Event as E
WHERE PE.Product_ID = Pr.Product_ID and
      PE.Event_ID = E.Event_ID and
      E.Event_Date like “*2016” and
      O.Order_ID = OD.Order_ID and
     OD.Product_ID = Pr.Product_ID and
     Pr.Product_Type_ID = PT.Product_Type_ID and
     PT.RecipeID = R.RecipeID and
     R.Ingredient_ID = Ing.Ingredient_ID and
     Ing.Purchase_ID = P.Purchase_ID and
     PP.Event_ID=PE.Event_ID;
```

4. Product Demand Analysis

This query will help our client to keep track of popular products, which can be used to determine the allocation of part-time employees' shifts. It will return the fluctuation of product demand by type and amount in each day/week.

- Extracts most popular products by adding up quantity

```
SQL> SELECT PT.Product_Name, sum(OD.Quantity) as Total_Amount
     FROM Product_Type PT, Product1 P, Order1 O, Order_Details OD
     WHERE PT.Product_Type_ID=P.Product_Type_ID and P.Order_ID=O.Order_ID and
           O.Order_ID=OD.Order_ID
     GROUP BY PT.Product_Name;
```

- The following query will give us number of products purchased per day. Then we plot this data into R for time series analysis.

```
SQL> SELECT O.Date_Placed, sum(OD.Quantity)
     FROM Product_Type PT, Product1 P, Order1 O, Order_Details OD
     WHERE PT.Product_Type_ID=P.Product_Type_ID and P.Order_ID=O.Order_ID and
           O.Order_ID=OD.Order_ID
     GROUP BY O.Date_Placed;
```

```
SQL> SELECT O.Date_Placed, O.Time_Placed, sum(OD.Quantity)
     FROM Product_Type PT, Product1 P, Order1 O, Order_Details OD
     WHERE PT.Product_Type_ID=P.Product_Type_ID and P.Order_ID=O.Order_ID and
           O.Order_ID=OD.Order_ID
     GROUP BY O.Date_Placed, O.Time_Placed;
```

5. Employee Scheduling

Use this for juice specialist scheduling to minimize the cost. Use Ampl

```
SQL> SELECT Juice_Specialist.*, Packer.*
     FROM Juice_Specialist INNER JOIN Packer ON Juice_Specialist.EID=Packer.EID
     UNION ALL
     SELECT Juice_Specialist.*,Packer.*
     FROM Juice_Specialist LEFT JOIN Packer ON Juice_Specialist.EID=Packer.EID
     UNION ALL
     SELECT Juice_Specialist.*,Packer.*
     FROM Juice_Specialist RIGHT JOIN Packer ON Juice_Specialist.EID=Packer.EID;
```

VII. Modeling and Analysis

Technique

The models we used in the analysis including P-center, P-median and Set Covering model for location planning based on geographical analysis and interactive mapping. Contaminated product query relies on linear regression for trend detecting. Both linear regression and visualization are involved in the marketing analysis. And the interactive Shiny app is provided in the link. For demand forecasting, frequency analysis and time series analysis

Comment [1]: I listed the techniques here. Please add things that are not mentioned here. Same for the software part.

are applied using seasonal arima model. Linear programming formulation is used for employee scheduling and sensitivity analysis is also provided.

Software

Tableau, R Studio and ampl.

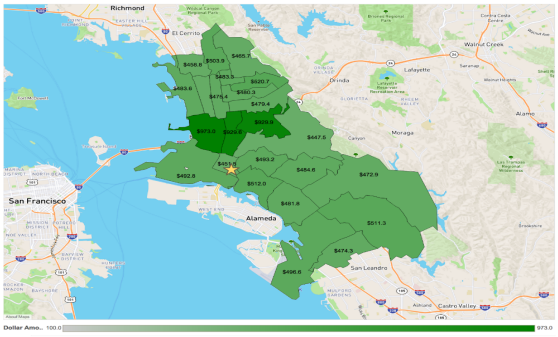
Query 1

A. Overview

In order to optimally answer the question of where Uptown Juice Co. should open up additional locations we took a two step approach. The first step involved importing the zip code and spending data into Tableau and then combining and plotting on a map in order to obtain a visual representation of where the demand for our client's products is coming from. The second step of our approach consisted in using the demand data for every zipcode obtained through the SQL code and fitting two location planning models (Capacitated P-Median and Capacitated P-Center) in order to concretely determine which are the optimal locations for our client to open additional stores.

B. Graphical Representation of Geographical Demand

After being extracted through SQL, zip code and spending data was then imported into Tableau (data analytics software) in order produce a graphical representation of where demand for our client's products is comer from. The following map was created by combining zip code data with spending-per-order data:



As we can see from the map, demand for Uptown Juice Co. Products is mainly concentrated in three regions near where the store is located.

Figure 1.1 Daily Dollar Amounts by Zip Code

C. Models

In order to concretely identify the optimal locations for additional Uptown Juice Co. stores, the spending per zip code obtained by generating the graphical visualization in Tableau was then utilized to construct a capacitated p-median model and a capacitated p-center model. In both models, spending per zip code was used to represent demand. For simplification purposes, each zip code was treated as a separate demand node and the distance between each zip code zone was considered to be the distance between demand nodes. Due to limited data availability and in order to facilitate the model, we assumed that stores do not have a limit on how much demand they can satisfy. In order to construct a more realistic model, we would need to conduct an in depth analysis of how the current store functions and how much demand it can supply.

Furthermore, since Uptown Juice Co. currently only has one location, we determined the p-value to be 2 given their current expansion capabilities.
Note: See appendix for demand per node and distance between nodes (figures 1 and 2)

Capacitated P-Median Model

The objective of this model is to place p stores in order to minimize the demand-weighted average distance between a demand node and the location in which a store was placed.

```
param dis{i in 1..24, j in 1..24};
param demand{i in 1..24};
param p=2;
param c = 40000;

var x {i in 1..24, j in 1..24} >= 0, <=1;
var y{j in 1..24} binary;

minimize Total_cost: sum{i in 1..24, j in 1..24}dis[i,j]*x[1,j];

subject to demandconstraint{i in 1..24}: sum{j in 1..24} x[i,j];
subject to prevent {j in 1..24}: sum{i in 1..24} demand[i]*x[i,j] <= c*y[j];
subject to facilities: sum{j in 1..24}y[j] <= p;

display y;
```

```
data;  ########## DATA STARTS HERE ###########
param dis: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 16 19 20 21 22 23 24 :=
      1 0 1.6 3.7 3.2 1.8 3.8 5.4 4.5 2.7 3.4 3.3 4.8 1.6 2.3 6.9 6.5 6.5 5.7 8.7 8.8 8.2 7.3 7.4 6.7
      2 1.6 0 4.7 3.8 1.9 3.8 4.7 3.6 2.1 2 3.1 3.4 1.5 3.5 6.9 5.4 5.3 4.4 7.7 7.6 6.9 6.1 6.5 5.3
      3 3.7 4.7 0 1.6 5.5 7.4 9 8.1 6.4 6.7 6.9 8 3.4 1.4 10.5 10 10 9.1 12.3 12.3 11.6 10.8 11.1 10
      4 3.2 3.8 1.6 0 4.9 7 8.3 7.3 5.6 5.7 6.4 7 2.2 1.6 9.6 9.1 8.9 8 11.3 11.1 10.4 9.7 10.2 8.9
      5 1.8 1.9 5.5 4.9 0 2.1 3.6 2.8 1.1 2.1 1.5 3.3 3 4.1 5.2 4.9 5 4.3 7.1 7.3 6.8 5.8 6.7 5.3
      6 3.8 3.8 7.4 7 2.1 0 2.2 2.2 1.9 3.2 0.7 3.3 5.1 6.1 4.1 4 4.4 4 5.9 6.4 6.2 5.1 4.4 5
      7 5.4 4.7 9 8.3 3.6 2.2 0 1.2 2.7 4 2.2 3.7 4 6.2 2.2 4.5 3.7 3.3 3.8 4.5 4.8 3.5 2.7 3.4
      8 4.5 3.6 8.1 7.3 2.9 2.2 1.2 0 2.1 2.4 1.8 1.5 6 7.5 2.4 2.2 2.8 2.2 4.2 4.5 4.5 3.1 3.5 2.8
      9 2.7 2.1 6.4 5.8 1.1 1.9 2.7 2.1 0 2.1 6.2 3.4 1.5 5.5 4.5 4.2 4.2 3.5 6.2 6.5 6.3 4.9 5.6 4.4
      10 3.4 2 6.7 5.7 2.3 3.2 4 2.4 2 0 3.2 1.2 4.3 6.7 4.1 3.7 2.8 2.2 6.5 6.4 5 3.9 5.4 3.3
      11 3.3 3.1 6.9 6.4 1.5 0.7 2.2 1.8 1.6 3.2 0 2.9 5.5 6 3.9 3.9 4.6 3.9 5.8 6.3 6.3 4.2 4.8 4.6
      12 4.6 3.4 8.7 7 3.3 3.3 3.9 3.1 5 2.3 1.2 2.9 0 5.4 7.7 2.9 2.4 1.9 11.4 4.4 4.4 2.8 4.2 2.2
      13 1.6 1.8 3.4 2.2 3 3.6 1.7 4.6 4.1 4.3 5.5 5.4 0 4.2 8.3 7.9 7.1 7.4 9.8 9.7 9.2 8.2 9.5 7.6
      14 2.3 3.5 1.4 1.6 4.1 6.1 8.2 7.5 5.5 6.7 6 4.2 0 9.9 8.7 9.7 8.9 11.7 12 11.7 10.4 10.8 9.9 2.1
      15 6.9 5.9 10.5 9.6 5.2 4.1 2 2.4 4.5 4.1 3.9 2.9 8.3 9.9 0 0.6 2.3 2.4 1.8 2.5 2.8 1.6 1.2 1.9
      16 6.5 5.4 10 9.1 4.9 4 2.4 2.2 4.2 3.7 3.9 2.4 7.9 9.7 0.6 0 1.7 1.8 2 2.3 2.5 1.1 1.8 1.2
      17 6.5 5.3 10 8.9 5 4.4 3.7 2.8 4.2 2.8 4.6 1.9 7.1 9.7 2.3 1.7 0 0.8 3 2.6 2.1 1.2 3.4 0.6
      18 5.7 4.4 9.1 8 4.3 4 3.3 2.2 3.5 2.2 3.9 1.1 7.4 8.9 2.4 0.8 0 3.5 3.3 2.9 1.8 3.6 1.1 2.1
      19 8.7 7.7 12.3 11.3 7.1 6.9 3.8 4.2 6.2 5.5 5.8 4.4 9.8 11.7 1.8 2 3 3.5 0 1.1 1.9 1.8 1.4 2.4
      20 8.8 7.6 12.3 11.1 7.3 6.4 4.5 4.5 6.5 5.4 6.3 4.4 9.7 12 2.5 2.3 2.6 3.3 1.1 0 0.9 1.6 2.5 2.2
      21 8.2 6.9 11.6 10.4 6.8 6.2 4.8 4.5 6.3 5 6.3 4 9.2 11.7 2.8 2.5 2.1 2.9 1.9 0.9 0 1.4 3.2 1.8
      22 7.3 6.1 10.8 9.7 5.8 5.1 3.5 3.1 4.9 3.9 4.2 2.8 8.2 10.4 1.6 1.1 1.2 1.8 1.8 1.6 1.4 0 2.4 0.66
      23 7.4 6.5 11.1 10.2 5.7 4.4 2.7 3.5 5.6 5.4 4.8 4.2 9.5 10.8 1.2 1.9 3.4 3.6 1.4 2.5 3.2 2.4 0 2.9
      24 6.7 5.3 10 8.9 5.4 5 3.4 2.8 4.4 3.3 4.6 2.2 7.6 9.9 1.9 1.2 0.6 1.1 2.4 1.8 0.7 2.9 2.1 0;

param demand:= 481 484 474 511 512 492 973 929 493 447 451 929 472 296 460 475 480 479 458 503 465 483 483 520;
```

AMPL Code:

AMPL Output:

Therefore, according to this model the optimal locations for Uptown Juice Co. to place additional stores in order to minimize the demand-weighted average distance between demand demand nodes and the stores would be nodes 2 and 8, which in term correspond to zip codes 94602 and 94609 (see appendix).

```
y [*] :=
 1 0   4 0   7 0   10 0   13 0   16 0   19 0   22 0
 2 1   5 0   8 1   11 0   14 0   17 0   20 0   23 0
 3 0   6 0   9 0   12 0   15 0   18 0   21 0   24 0
;
```

Capacitated P-Center Model

The objective of this model is to place p stores in order to minimize the maximum distance between any demand node and the location in which a store was placed. In this model, there are no capacity constraints at the facilities.

AMPL Code:

```
param dis{i in 1..24, j in 1..24};
param demand{i in 1..24};
param p = 2;

var x {i in 1..24, j in 1..24}>=0,<=1;
var y {j in 1..24} binary;
var z >= 0;

minimize Total_distance: z;

subject to demconstraint {i in 1..24}: sum {j in 1..24} x[i,j] = 1;
subject to prevent {i in 1..24, j in 1..24}: x[i,j] <= y[j];
subject to facilities: sum {j in 1..24} y[j] <= p;
subject to disvar {i in 1..24}: sum {j in 1..24} dis[i,j] * x[i,j] <= z;

display y;
```

```
data;  ############ DATA STARTS HERE ############
param dis: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 :=
      1 0 1.6 3.7 3.2 1.8 3.8 5.4 4.5 2.7 3.4 3.3 4.6 1.6 2.3 6.9 6.5 6.5 5.7 8.7 8.8 8.2 7.3 7.4 6.7
      2 1.6 0 4.7 3.8 1.9 3.8 4.7 3.6 2.1 2 3.1 3.4 1.5 3.5 5.9 5.4 5.3 4.4 7.7 7.6 6.9 6.1 6.5 5.3
      3 3.7 4.7 0 1.6 5.5 7.4 9 8.1 6.4 6.7 6.9 8 3.4 1.4 10.5 10 10 9.1 12.3 12.3 11.6 10.8 11.1 10
      4 3.2 3.8 1.6 0 4.9 7 8.3 7.3 5.6 5.7 6.4 7 2.2 1.6 9.6 9.1 8.9 8 11.3 11.1 10.4 9.7 10.2 8.9
      5 1.8 1.9 5.5 4.9 0 2.1 3.6 2.9 1.1 2.1 1.5 3.3 3 4.1 5.2 4.9 5 4.3 7.1 7.3 6.8 5.8 6.7 5.3
      6 3.8 3.8 7.4 7 2.1 0 2.2 2.2 1.9 3.2 0.7 3.3 5.1 6.1 4.1 4 4.4 4 5.9 6.4 6.2 5.1 4.4 5
      7 5.4 4.7 9 8.3 3.6 2.2 0 1.2 2.7 4 2.2 3.7 4 8.2 2 2.4 3.7 3.3 3.8 4.5 4.8 3.5 2.7 3.4
      8 4.5 3.6 8.1 7.3 2.9 2.2 1.2 0 2.1 2.4 1.8 1.5 6 7.5 2.4 2.2 2.8 2.2 4.2 4.5 4.5 3.1 3.5 2.8
      9 2.7 2.1 6.4 5.6 1.1 1.9 2.7 2.1 0 2.1 6 2.3 4.1 5.5 4.5 4.2 4.2 3.5 6.2 6.5 6.3 4.9 5.6 4.4
      10 3.4 2 6.7 5.7 2.3 3.2 4 2.4 2 0 3.2 1.2 4.3 6.7 4.1 3.7 2.8 2.2 5.5 5.4 5 3.9 5.4 3.3
      11 3.3 3.1 6.9 6.4 1.5 0.7 2.2 1.8 1.6 3.2 0 2.9 5.5 6 3.9 3.9 4.6 3.9 5.8 6.3 6.3 4.2 4.8 4.6
      12 4.6 3.4 8 7 3.3 3.3 3 1.5 2.3 1.2 2.9 0 5.4 7.7 2.9 2.4 1.9 1.1 4.4 4.4 4 2.8 4.2 2.2
      13 1.6 1.5 3.4 2.2 3 5.1 7.4 6 4.1 4.3 5.5 5.4 0 4.2 8.3 7.9 7.1 7.4 9.8 9.7 9.2 8.2 9.5 7.6
      14 2.3 3.5 1.4 1.6 4.1 6.1 8.2 7.5 5.5 6.7 6 4.2 0 9.9 9.7 9.7 8.9 11.7 12 11.7 10.4 10.8 9.9 2.1
      15 6.9 5.9 10.5 9.6 5.2 4.1 2 2.4 4.5 4.1 3.9 2.9 9.3 9.9 0 0.6 2.3 2.4 1.8 2.5 2.8 1.6 1.2 1.9
      16 6.5 5.4 10 9.1 4.9 4 2.4 2.2 4.2 3.7 3.9 2.4 7.9 9.7 0.6 0 1.7 1.8 2 2.3 2.5 1.1 1.8 1.2
      17 6.5 5.3 10 8.9 5 4.4 3.7 2.8 4.2 2.8 4.6 1.9 7.1 9.7 2.3 1.7 0 0.8 3 2.6 2.1 1.2 3.4 0.6
      18 5.7 4.4 9.1 8 4.3 4 3.3 2.2 3.5 2.2 3.9 1.1 7.4 8.9 2.4 0.8 0 3.5 3.3 2.9 1.8 3.6 1.1 2.1
      19 8.7 7.7 12.3 11.3 7.1 5.9 3.8 4.2 6.2 5.5 5.8 4.4 9.8 11.7 1.8 2 3 3.5 0 1.1 1.9 1.8 1.4 2.4
      20 8.8 7.6 12.3 11.1 7.3 6.4 4.5 4.5 6.5 5.4 6.3 4.4 9.7 12 2.5 2.3 2.6 3.3 1.1 0 0.9 1.6 2.5 2.2
      21 8.2 6.9 11.6 10.4 6.8 6.2 4.8 4.5 6.3 5 6.3 4 9.2 11.7 2.8 2.5 2.1 2.9 1.9 0.9 0 1.4 3.2 1.8
      22 7.3 6.1 10.8 9.7 5.8 5.1 3.5 3.1 4.9 3.9 4.2 2.8 8.2 10.4 1.6 1.1 1.2 1.8 1.8 1.6 1.4 0 2.4 0.66
      23 7.4 6.5 11.1 10.2 5.7 4.4 2.7 3.5 5.6 5.4 4.8 4.2 9.5 10.8 1.2 1.8 3.4 3.6 1.4 2.5 3.2 2.4 0 2.9
      24 6.7 5.3 10 8.9 5.4 5 3.4 2.8 4.4 3.3 4.6 2.2 7.6 9.9 1.9 1.2 0.6 1.1 2.4 1.8 0.7 2.9 2.1 0;

param demand:= 481 484 474 511 512 492 973 929 493 447 451 929 472 296 460 475 480 479 458 503 465 483 483 520;
```

```
y [*] :=
 1 1   4 0   7 0   10 0   13 0   16 0   19 0   22 0
 2 1   5 0   8 0   11 0   14 0   17 0   20 0   23 0
 3 0   6 0   9 0   12 0   15 0   18 0   21 0   24 0
;
```

AMPL Output:

Therefore, according to this model the optimal locations for Uptown Juice Co. to place additional stores in order to minimize the demand-weighted average distance between demand demand nodes and the stores would be nodes 1 and 2, which in term correspond to zip codes 94601 and 94602 (see appendix). It is important to note that results for this model differ from those obtained from the previous model given that they have different objective functions.

Query 2

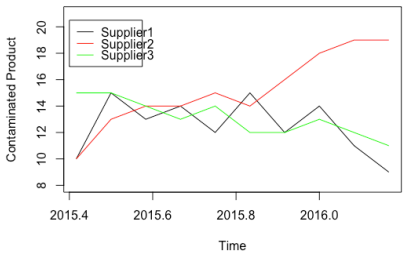
A. Overview

There are two parts of the query. One is to use SQL to extract information of contaminated product and customer who have bought the same batch of product in order to contact these customers to minimize the damage. The other part is to find the ingredients being used in these contaminated products and trace back to suppliers that we purchase these ingredients from. Plotting this data in R on a time domain would allow us to see if there exists a trend from a particular supplier .

Part 1 can be accomplished using SQl only, and the company would assign a designated person to contact all these customers through phone or email to let them know what the situation is and provide them with refund or future discounts.

B. Visualization and Model

Part 2 will need visualization and linear regression. Below is the time series plot on quantity purchased from each supplier. This visualization would be enough to spot the trend, but by applying a linear regression, we would be able to get more detailed information.




```
Call:
lm(formula = contamsup1 ~ time(contamsup1))

Residuals:
    Min      1Q  Median      3Q     Max
-3.4545 -0.7197 -0.1061  1.6136  2.6061

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      5143.606   5553.527   0.926    0.381
time(contamsup1)   -2.545      2.755  -0.924    0.383

Residual standard error: 2.085 on 8 degrees of freedom
Multiple R-squared:  0.09642,   Adjusted R-squared:  -0.01653
F-statistic: 0.8537 on 1 and 8 DF,  p-value: 0.3825
```

```
lm(formula = contamsup2 ~ time(contamsup2))

Residuals:
    Min      1Q  Median      3Q     Max
-1.6485 -0.4682  0.1970  0.6349  1.0424

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      -21682.048   2530.761  -8.567 2.66e-05 ***
time(contamsup2)     10.764      1.255   8.573 2.64e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9503 on 8 degrees of freedom
Multiple R-squared:  0.9018,    Adjusted R-squared:  0.8896
F-statistic:  73.5 on 1 and 8 DF,  p-value: 2.644e-05
```

```
lm(formula = contamsup3 ~ time(contamsup3))

Residuals:
     Min       1Q   Median       3Q      Max
-0.89697 -0.43636 -0.02121  0.43939  0.91515

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      9835.5030  1709.6704   5.753 0.000428 ***
time(contamsup3)   -4.8727     0.8481  -5.745 0.000431 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.642 on 8 degrees of freedom
Multiple R-squared:  0.8049,    Adjusted R-squared:  0.7805
F-statistic: 33.01 on 1 and 8 DF,  p-value: 0.0004314
```

Among these supplies, only supplier 2 has a positive coefficient, which means supplier 2 has an upper trend on the quantity purchased of ingredients that caused contaminated products. This will be later on reported to the company and the purchase department will look into this and find out what the issues are.

We believe that this should not be a major issue for the company, but since these ingredients and products are highly perishable, it is good to have this query available. Once a problem is reported by one customer, all the products that were made on the same day will can be found. The employees who were involved in the production can also be tracked; the suppliers could be tracked and all the corresponding purchased could be tracked. This will allow the problem to be solved in minimal time.

Query 3: Marketing Analysis

A. Overview

For this analysis, We aim to help Uptown Juice Company better determine each product's discount rate and its corresponding period for the discount during each product promotion event. From which they should be able to get benefits such as a better solution to deal with

perishability; better understanding relationship between discount and profit per product; and making quicker promotion decisions via the app, etc.

B. Models

First, we help determine how many days our client needs to sell each product out during promotion. We aim to find the most reasonable durations so that our client will not start promotion too early or too late. By SQL query, we extract all promotion information for each product during year 2016 (Appendix Query 3 Data from SQL). We plot the following box plot (Figure 3.1) using R programming language to show the distribution of promotion duration for each product. We also construct a simple linear regression in R to confirm our observations from boxplot. We find a relationship between product type and duration. Here we have eight products. As a categorical variable, each level of $product\ type$ become a boolean feature. That is, $duration\ = a_1x_1\ + a_2x_2 + \ldots + a_8x_8$ where $x_i$is a boolean variable (1 or 0) representing if a product is type $i$and $a_i$is corresponding coefficient. The output of linear model is shown on the right (Figure 3.2).

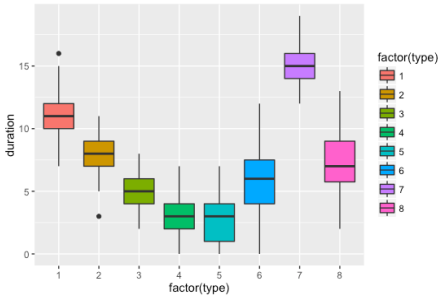


Figure 3.1

```
Call:
lm(formula = duration ~ type, data = duration_by_type)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9091 -0.9888  0.0112  1.0112  6.0909

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.98876    0.09599  114.47   <2e-16 ***
type2       -2.94671    0.14391  -20.48   <2e-16 ***
type3       -6.07700    0.18258  -33.28   <2e-16 ***
type4       -8.16598    0.15744  -51.87   <2e-16 ***
type5       -8.31080    0.22564  -36.83   <2e-16 ***
type6       -5.07967    0.48257  -10.53   <2e-16 ***
type7        3.91401    0.20829   18.79   <2e-16 ***
type8       -3.78288    0.21306  -17.75   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.569 on 943 degrees of freedom
Multiple R-squared:  0.8459, Adjusted R-squared:  0.8448
F-statistic: 739.5 on 7 and 943 DF,  p-value: < 2.2e-16
```

Figure 3.2

In order to find the most reasonable discount which maximized promotion profit for each product, we first formulate an optimization problem for each product as below:$max_{discount} Promotion\ Profit = (Unit\ Price \times (1 - discount)\ - Unit\ Cost) \times Quantity\ Sold$or $max_d \pi = (p \times (1 - d) - c) \times q$where π is profit, p is unit price, d is discount, c is unit cost, and q is quantity sold. The promotion profit for each product is defined to be a function of unit price, discount, unit cost and quantity sold. Here, quantity_sold can be written as a function of discount because there is a linear relationship between discount and quantity_sold for each product. For example, the plots (Figure 3.3) below demonstrates a linear relationship between discount and quantity sold for Almond Milk, Soy Milk, Apple Juice and Veggie Juice. The blue line is a fitted line which show the expected quantity sold for each discount for each product. Therefore, the optimization problem can be re-written as $max_d \pi = (p \times (1 - d) - c) \times (a \times d\ + b)$, where a is the slope and b is intercept for the fitted line. The profit function is concave, so we can obtain a maximum. A discount d maximizes the promotion profit when $\frac{\partial \pi}{\partial d} = pa - 2pad - pb - ca = 0 \Rightarrow d\ = \frac{1}{2}\left(1 - \frac{b}{a} - \frac{c}{p}\right)$.

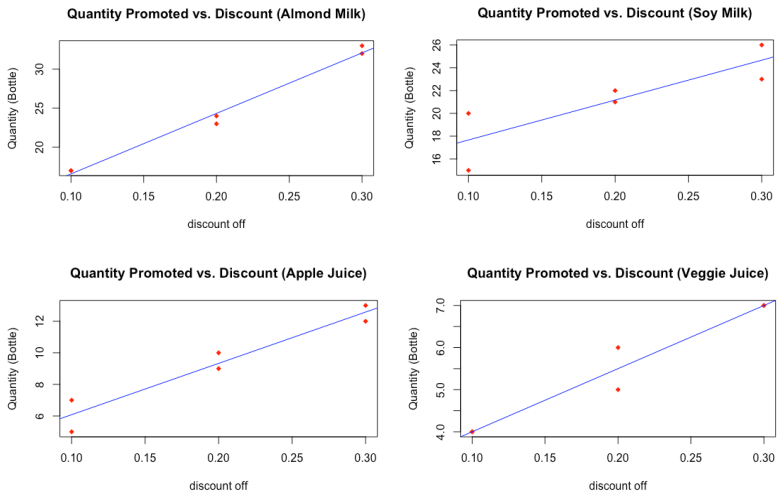


Figure 3.3

C. Results Analysis

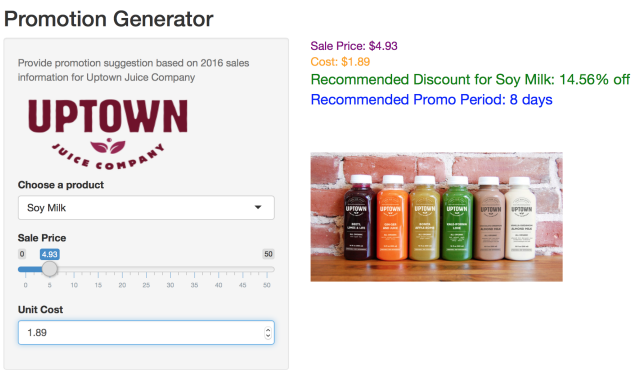
Based on promotion details in 2016, we find out an expected duration for each product. For product 1-8, expected durations are 11, 8, 5, 3, 3, 5, 15, and 7 days respectively. The observation from the box plot (Figure 3.1) and output of linear model (Figure 3.2) coincide. The output of linear model shows that p-values of all features are significant and $R^2$is about 0.85. About 85% of variation in $duration$ can be explained by $product\ type$. We would recommend our client to accept the expected durations as their future promotion duration.

For discount, after solving the optimization problem, we obtain $d = \frac{1}{2}\left(1 - \frac{b}{a} - \frac{c}{p}\right)$. The optimal discount depends on two ratios: ratio of intercept and slope of linear regression line of plot of quantity sold and discount and ratio of unit cost and unit price. From the expression of d, discount will not be greater than 50 percent because two ratios are positive. When the client has a more than 50 percent off sale, our client will not achieve a maximum profit. $\frac{b}{a}$is fixed for each product, so one product can have a higher discount if unit price is much higher than unit cost.

D. Shiny App

We implement our recommendations about duration and discount in Shiny App, which is an interactive web application. Following is the link for our promotion generator app and a screenshot shown as an example. By easily selecting a product type and input of the cost and sales price for that product, there will appear a recommended discount rate and period for that product on the left hand side of the screen. Furthermore, this app could help our client better decide on the cost and price for existed products when in the future there is a change of cost for some ingredient in the market and they can get some recommendation on how to adjust the sales price to get a better discount rate as future use. Detailed implementation of the Shiny App can be found in Appendix.

Shiny App: https://liuanna.shinyapps.io/promotion_app/



E. Future Plan

At this moment, the data we used for the marketing analysis is what we made based on the products sold at the store. In the future, when more data from the client are available, we can build more accurate models for this system. Also we plan to connect our database with R, making it a real time simulator and a more convenient and less time-consuming app to meet the demand for expanded dataset from the client.

Query 4 Demand Forecasting

A. Overview

For this analysis, our goal is to find out the fluctuation of the sales amount on a monthly basis as well as daily basis. Both information are extracted using SQL and plotted in R for visualization. By applying time series analysis on both time and frequency domain, we would like to forecast the daily demand for the next month. This forecasted demand should help them make decisions on how much to purchase from each supplier as well as the inventory level. Mostly importantly, the forecasted demand should determine the scheduling of the employees.

B. Monthly Sales Frequency Analysis and Model Fitting

Figure 1 is a time series plot on monthly sales for the past three years with monthly plotting symbols added. Peaks occur mostly in August and the lowest points are in January, February or March. Slope is generally increasing from February to August and starting to drop from August to February of next year.

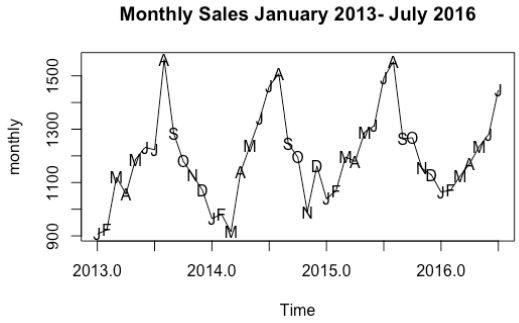


Figure 1

Next step is to detrend and apply transformation on time series if needed. Figure 2 indicates that there exists a linear trend although the bar indicates that this linear trend is not very significant. Since the number of observations is very limited, we do consider to take out this linear trend first and start with time series analysis on the residuals. Below is the result of linear regression.

```
Call:
lm(formula = monthly ~ time(monthly))

Coefficients:
  (Intercept)  time(monthly)
     -83012.7           41.8
```

Figure 2 is the sample autocorrelation function on the time series after taking out the linear trend. We can clearly see that the MA part has a seasonality of 6 months. However, the partial autocorrelation function (Figure 4) indicates that the AR part of the time series do not have a seasonality. So to determine the seasonality of the time series as a whole. We plotted a periodogram shown in FIgure 5. We added a red vertical bar at 1/12 to verify his peak frequency occurs every 12 months. Although there is variation every quarter and 6 months, the significant frequency is still every 12 months. But in terms of future decisions, quarterly and 6-month data should still be considered to look for necessary patterns.

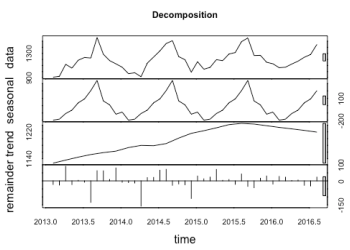


Figure 2

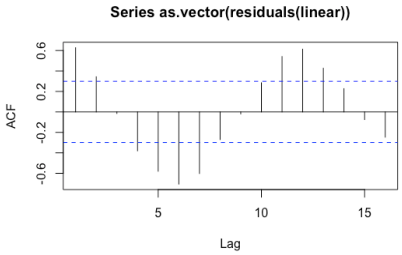
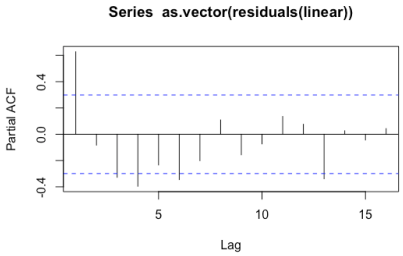


Figure 3



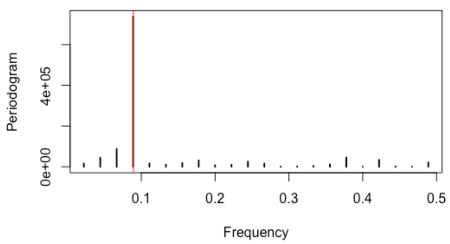
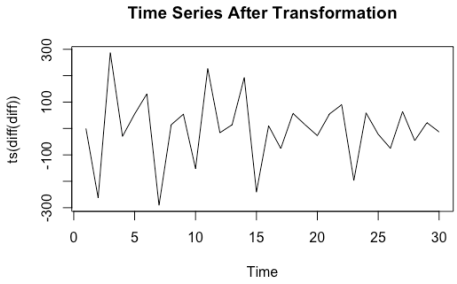


Figure 4 Figure5

Then we applied a seasonal differencing of lag 12 transformation on the residuals to to take out this seasonal trend. However, the time series does not seem to fluctuate around a constant mean value. To make sure the time series becomes stationary, we again applied a regular differencing of lag 1. Below is the time series plot after decomposition and transformation, which seems to be a stationary time series. Since there is not a good way to verify the stationarity of time series, this part of the analysis is highly subjective. But this is unavoidable on analysis that is highly dependant on graphics. Then we looked at sample autocorrelation function (acf), sample partial autocorrelation function (acf) and extended autocorrelation function (eacf) to verify what the parameters are. We also tried auto.arima command to double check on our parameters of p,d,q and P, D, Q. Since auto.arima gives the best parameters based on AIC(Akaike Information Criterion) and BIC (Bayesian Information Criterion), so we do not need to manually check these two criterions again.



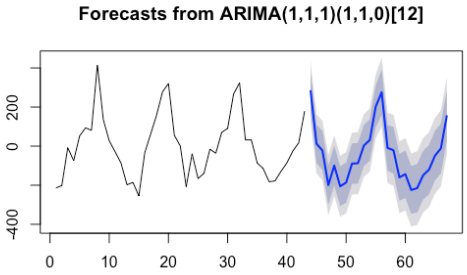
Below is the result of fitting this seasonal arima model, and the model is expressed below.

```
Call:
arima(x = residuals(linear), order = c(1, 1, 1), seasonal = list(order = c(1,
    1, 0), period = 12))

Coefficients:
         ar1      ma1     sar1
      0.1012  -0.8847  -0.6498
s.e.  0.2090   0.1577   0.1433

sigma^2 estimated as 5644:  log likelihood = -176.25,  aic = 358.49
```

$$(1 - 0.1012B)(1 + 0.6498B^{12})(1 - B) \times Y_t = (1 - 0.8847B) \times e_t$$

C. Monthly Demand Forecasting

With this information on hand, we used the forecast command that is built in the forecast package and was able to get the forecasted monthly demand. Note that the negative number exists because the linear trend was taken out the first step. This was something that we did not consider before the presentation. So both the seasonal arima model and the forecasted demand plot is slightly different. We thought because the linear trend was not significant based on the decomposition plot, so it should not be considered in the analysis. However, with this number of analysis, any trend should be captured in the model to make sure we can get a better accuracy.



Below is a forecasted monthly demand for the next 6 months. We would use month 44 which is August 2016 as an illustration. This information could also be passed on to the company for them to check whether this number is close enough to what the actual demand is in August.

```
> forecast(fit1, h=6)
   Point Forecast      Lo 80      Hi 80      Lo 95     Hi 95
44      281.87592  185.54289  378.20895  134.5473 429.20457
45       12.38553  -86.18964  110.96069 -138.3722 163.14324
46      -22.28514 -121.74321   77.17293 -174.3931 129.82286
47     -199.19472 -299.42795  -98.96150 -352.4882 -45.90123
48      -99.66033 -200.65303    1.33238 -254.1153  54.79469
49     -205.48121 -307.22676 -103.73567 -361.0876 -49.87483
```

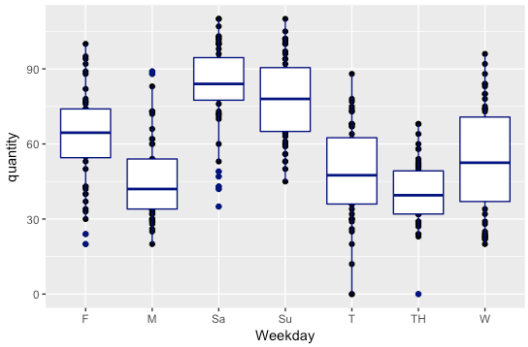
We first need to add the linear trend and then add this forecasted demand. Below is what the demand turned out to be for August 2016, which is 1503 since it needs to be an integer.

```
> num<-2016.667 * 41.8 - 83012.7+218.875921
> num
[1] 1502.857
```

The 95 % interval would give us [1419, 1714].

E. Daily Sales Demand Forecasting

Using SQL, we were able to get the daily demand, and plot the yearly demand from July 2015 to July 2016. Below is a scatter plot and box plot. This helps to visualize the fluctuation of the daily Demand. Most sales are generated on weekends, especially Saturday, and Sunday has the second largest demand. Friday also has a relatively high demand. Sales on weekdays are less than weekends in general, but Wednesday seem to have higher mean value than other weekdays. Among all the days, Wednesday has the highest variance, and Sunday also has a very high variance. The predictions on these two days might be off since the high variance. Tuesday also has a high variance but because the outliers. Several big holidays are on Tuesdays, which has affected the sales information.



Based on different needs on each day of the week, we have assigned different weights on each day, which are 0.1387, 0.1443, 0.1604, 0.1224, 0.1870, 0.2472, 0.2348 for Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday respectively. And the table below contains the forecasted daily demand

	Forecasted Daily Demand	95% Confidence Interval
Monday	53	[50, 60]
Tuesday	55	[52, 62]
Wednesday	61	[57, 69]
Thursday	47	[44, 53]
Friday	71	[67, 81]
Saturday	93	[88, 106]

Sunday	89	[84, 101]

F. Recommendation and Future Plan

The table in part E contains the recommended demand on each day of the week in August 2016. The data is generated by us according to reasonable assumptions, but is not the real data from the company. We have inserted about 10 to 15 entries in each table in our access file, but considering the data needed for time series analysis, we had to simulate some data in addition to what we inserted. Once we have the real data, the time series model will need to be adjusted, but the overall approach should remain the same. Our goal is to have this forecasted August demand on hand to compare with the real demand in August to test the model. This information will be useful to help with employee scheduling that will be discussed in the next section.

Query 5 Employee Scheduling

A. Overview

In this query, we want to answer the question "Can we figure out the optimal schedule for juice specialists and packers based on the forecasted demand?" In order to set the employee schedule, we will construct an optimization problem, which is to determine the amount of hours each employee will work for all days in a given week, minimizing the company's total wage expenses, subject to many constraints. The optimization problem is implemented in Ampl.

B. The Optimization Problem

Decision Variables:

x_{ij}: i'th juice specialist's number of working hours at day j, where i=1,2,...,10, we have 10 juice specialists, j=1,2,...,7, Mon to Sun

E_{ij}: i'th juice specialist's number of overtime hours accumulated from day 1 to day j

y_{hj}: h'th packer's number of working hours at day j, where h=1,2,...,5, we have 5 packers, j=1,2,...,7, Mon to Sun

F_{hj}: h'th packer's number of overtime hours accumulated from day 1 to day j

Parameters:

D_j:Demand for day j

a_i:Quantity of juice produced per hour by juice specialist i

b_h:Quantity of juice packed per hour by packer h

w_i:Hourly wage of worker i

c_i:Extra wage for each additional hour higher than part-time threshold(20 hours per week).

Assumptions:

Under our assumptions, the working schedule is set without considering employee's preference on when to work. Such consideration could be add later using more advanced methods.

Under our assumptions, juice specialists and packers should be hired as part-time employees. Standard wage is paid to them when they work less than or equal to 20 hours per week. If they

work 20(exclusive) to 40(inclusive) hours per week, additional wage(a.k.a penalty wage, since it acts as a penalty to the company) is paid to them. They are not allowed to work more than 40 hours per week.

Objective Function:

$$min \sum_{j=1}^{7} \left(\sum_{i=1}^{10} (w_i x_{ij} + c_i E_{ij}) + \sum_{h=1}^{5} (\mu_h y_{hj} + d_h F_{hj})\right)$$

Note that $\sum_{i=1}^{10} (w_i x_{ij})$ is the sum of standard wage paid out to juice specialists on day j, and $\sum_{i=1}^{10} (c_i E_{ij})$is the sum of penalty wage paid out to juice specialists on day j.
Similarly, $\sum_{h=1}^{5} (\mu_h y_{hj})$ and $\sum_{h=1}^{5} (d_h F_{hj})$ represent standard and penalty wage paid out to packers on day j.

Constraints:

1. $\sum_{i=1}^{10} a_i x_{ij} = \sum_{h=1}^{5} b_h y_{hj}$ This constraint is a link between x and y. It shows the relationship between juice specialists and packers, i.e. amount of juice produced by juice specialists should be equal to amount of juice packed by packers on a given day.
2. $\sum_{j=1}^{7} x_{ij} \leq 40 \; \forall i, \sum_{j=1}^{7} y_{hj} \leq 40 \; \forall h$ These constraints disallow any juice specialist or packer to work more than 40 hours per week.
3. $E_{ik} \geq (\sum_{j=1}^{k} x_{ij} - 20) \; \forall k \; where \; k = 1,2,\ldots,7$, $F_{hk} \geq (\sum_{j=1}^{k} y_{hj} - 20) \; \forall k$, These two constraints make sure workers only receive extra (penalty) wage when they work more than 20 hours.
4. $E_{ik} \geq 0, F_{hk} \geq 0$ These two constraints make sure wage is not deducted from total expenses when a worker works less than 20 hours.
5. $\sum_{i=1}^{10} a_i X_{ij} \geq D_i \; \forall j$. This constraint makes sure the amount of juice produced each day meets the demand.Since previously, we already asid $\sum_{i=1}^{10} a_i x_{ij} = \sum_{h=1}^{5} b_h y_{hj}$, there is no need to make both left hand side and right hand wide =0.
6. $0 \leq x_{ij}, y_{hj}, E_{ij}, F_{h.j} \leq 8$ This constraint disallows negative value for decision variable, since negative hours does not make sense.

C. Results

The above optimization problem is implemented using Ampl. Here is a screenshot of Ampl.

```
param I; # The number of juice specialists
param J; # The number of days in a week
param H; # The number of packers

param a{i in 1..I}; # The quantity of juice produced per hour by specialist i
param b{h in 1..H}; # The quantity of juice packed per hour by packer h
param w{i in 1..I}; # The wage of specialist i
param m{h in 1..H}; # The wage of packer h
param c{i in 1..I}; # The wage for additional working hours of specialist i
param d{h in 1..H}; # The wage for additional working hours of packer h
param demand{j in 1..J}; # Demand can be changed by season

var x{i in 1..I, j in 1..J} >= 0, <= 8; # i th specialist's working hours on day j
var y{h in 1..H, j in 1..J} >= 0, <= 8; # h th packer's working hours on day j
var E{i in 1..I, j in 1..J} >= 0; # i th specialist's accumulative additional working hours until day j
var F{h in 1..H, j in 1..J} >= 0; # h th packer's accumulative additional woring hours until day j

minimize mincosts: sum{j in 1..J} (sum{i in 1..I} (w[i]*x[i,j] + c[i]*E[i,j]) + sum{h in 1..H} (m[h]*y[h,j] +d[h]*F[h,j]));

subject to SpecialistHoursLimit {i in 1..I}:sum{j in 1..J} x[i,j] <= 40;
subject to PackerHoursLimit {h in 1..H}:sum{j in 1..J} y[h,j] <= 40;
subject to specialistDay1{i in 1..I}:E[i,1] >= x[i,1] -20;
subject to specialistDay2{i in 1..I}:E[i,2] >= sum{j in 1..2}x[i,j] -20;
subject to specialistDay3{i in 1..I}:E[i,3] >= sum{j in 1..3}x[i,j] -20;
subject to specialistDay4{i in 1..I}:E[i,4] >= sum{j in 1..4}x[i,j] -20;
subject to specialistDay5{i in 1..I}:E[i,5] >= sum{j in 1..5}x[i,j] -20;
subject to specialistDay6{i in 1..I}:E[i,6] >= sum{j in 1..6}x[i,j] -20;
subject to specialistDay7{i in 1..I}:E[i,7] >= sum{j in 1..7}x[i,j] -20;
subject to packerDay1{h in 1..H}:F[h,1] >= y[h,1] -20;
subject to packerDay2{h in 1..H}:F[h,2] >= sum{j in 1..2}y[h,j] -20;
subject to packerDay3{h in 1..H}:F[h,3] >= sum{j in 1..3}y[h,j] -20;
subject to packerDay4{h in 1..H}:F[h,4] >= sum{j in 1..4}y[h,j] -20;
subject to packerDay5{h in 1..H}:F[h,5] >= sum{j in 1..5}y[h,j] -20;
subject to packerDay6{h in 1..H}:F[h,6] >= sum{j in 1..6}y[h,j] -20;
subject to packerDay7{h in 1..H}:F[h,7] >= sum{j in 1..7}y[h,j] -20;
subject to MakeJuice{j in 1..J}:sum{i in 1..I} (a[i]*x[i,j]) >= demand[j];
subject to NumProducedMatch{j in 1..J}:sum{i in 1..I}(a[i]*x[i,j])=sum{h in 1..H}(b[h]*y[h,j]);
```

The below table is optimal solution for minimizing the cost of employees using forecasted demands from query 4. That is {53, 55, 61, 47, 71, 93, 89} from monday to sunday.

	Juice Specialist Optimal Hours										*Packer Optimal Hours*				
Day	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**
Mon					4			4	2.6	8	4	8		7.1	
Tue						8		8		4.2	8	8	6		1
Wed						8		8		6.2	8		8		6.3
Thur							8		8	0.2				8	4.8
Fri	4	4	4	4	4		8		4.2			2.8	6	4.9	8
Sat	8	8	8	8	8		1.9		5.2			5.2	8	8	8
Sun	8	8	8	8	8	2.8	2.1			1.4		3.6	8	8	8
Total	20	20	20	20	24	18.8	20	20	20	20	20	27.6	36	36	36.1

Table. Optimal Schedule

:		y.down	y.current	y.up
1	1	12.25	12.25	12.25
1	2	-1e+20	12.25	12.25
1	3	-1e+20	12.25	12.25
1	4	12.25	12.25	1e+20
1	5	12.25	12.25	1e+20
1	6	12.25	12.25	12.25
1	7	7.3	12.25	15.55
2	1	-1e+20	13.25	13.25
2	2	-1e+20	13.25	13.25
2	3	13.25	13.25	1e+20
2	4	13.25	13.25	1e+20
2	5	13.25	13.25	13.25
2	6	13.25	13.25	13.25
2	7	13.25	13.25	19.4375
3	1	13.25	13.25	1e+20
3	2	13.25	13.25	13.25
3	3	-1e+20	13.25	13.25
3	4	13.25	13.25	1e+20
3	5	13.25	13.25	13.25
3	6	-1e+20	13.25	13.25
3	7	-1e+20	13.25	13.25
4	1	16.25	16.25	16.25
4	2	16.25	16.25	1e+20
4	3	16.25	16.25	1e+20
4	4	-1e+20	16.25	16.25
4	5	16.25	16.25	16.25
4	6	-1e+20	16.25	27.75
4	7	-1e+20	16.25	23.15
5	1	18.25	18.25	1e+20
5	2	18.25	18.25	18.25
5	3	18.25	18.25	18.25
5	4	18.25	18.25	18.25
5	5	-1e+20	18.25	18.25
5	6	-1e+20	18.25	33
5	7	-1e+20	18.25	27.1

Each cell represents how many hours a specific employee is assigned to work. This is obtained based on the ampl output. As shown in the last line of the schedule, most employees work exactly 20 hours in a week, which is the amount of hours we expect part-time workers to do. Some juice specialists are scheduled to work less than 20 hours, because comparing the wage paid out to him or her with his or her efficiency in making juice, it is not cost effective. On the other hand, some packers are scheduled to work over 20 hours, because we have limited number of packers available to meet the demand. Hence, some of the most cost-effective (high efficiency compared to their wage) packers are scheduled to work additional hours. This is the sensitivity analysis result for packers from Ampl. The first number is PackerID and the second number represents the days of the week: Monday is 1, Tuesday is 2 and so on. It shows how much each packer's wage can vary without changing our optimal solution on the above table. The y.down column and y.up column give the allowable range for wages to vary while keeping the optimal solution the same. For example, as long as packer 1's Sunday wage stays between $7.3 and $15.55, packer 1 will be required to work for 8 hours on Sunday.

VIII. Normalization

1. Supplier (SID, Company_Name, Street, City, State, Zip, Contact_Person, Phone, Email, Discount)

This relationship records all indicators related to suppliers. Email is considered as a multi-valued attribute since the database obtains at least one personal email and one company email. There is one designated contact person from the supplier's side. To normalize this relationship, considering from 1NF, 2NF, 3NF, and Boyce-Codd Normal Form (BCNF).

1NF: This original relation violates the first normal form because of the multi-valued attribute Email :

Supplier (SID, Company_Name, Street, City, State, Zip, Contact_Person, Phone, Discount)
Supplier_Email(SID, Email)

2NF: No partial dependency in this relationship, so at this step, the normalized relations remain the same.

Supplier (SID, Company_Name, Street, City, State, Zip, Contact_Person, Phone, Discount)
Supplier_Email(SID, Email)

3NF: SID determines Company_Name. Company_Name determines phone number and supplier's address (We have assumed each supplier has only one location. This is true in this case, because all the suppliers are local farms). Zip code could determine state. From the functional dependencies above, we know Company_Name, phone number, supplier's address are not primary keys, but there exists pairs (A,B) in non-prime attribute that A functionally determines B. So Supplier relationship violates the 3NF. Here is final result after normalization.

Supplier_Email(SID, Email)
Supplier_Info (SID, Company_Name, Discount)
Supplier_Contact_Person (SID, Contact_Person)
Contact_Person_Phone(SID, Phone)
Supplier_Address(SID, Street, City, Zip)
Supplier_State(Zip, State)

This final result after normalization does not violate BCNF, because no non-primary key could determine a primary key.

2. Employee (EID, Fname, Lname, MI, Emails, Phone, SSN, Street, City, State, Zip, Bdate, Job_Title[2], Salary)

This relationship records all indicators related to employees. Email is considered as a multi-valued attribute since the database obtains at least one personal email and one company email. To normalize this relationship, consider from 1NF, 2NF, 3NF, and BCNF.

1NF: First considering e-mails as a multi-value attribute :

Employee (EID, Fname, Lname, MI, Phone, SSN, Street, City, State, Zip, Bdate, Job_Title[2], Salary)
Employee_Email(EID, Email)

2NF: No partial dependency in this relationship, so at this step, the normalized relations remain the same.

Employee (EID, Fname, Lname, MI, Phone, SSN, Street, City, State, Zip, Bdate, Job_Title[2], Salary)
Employee_Email(EID, Email)

3NF: EID could determine a person including the combination of Fname, Lname, MI. Once a person is confirmed, his birthday, phone number, address and salary could be determined as well. Names, Birthday, phone number, address and salary are all non primary keys, so the relation violates 3NF, like pairs (A,B) in non-prime attribute that A functionally determines B. Here is the rewritten relationships after normalization.

Employee_Name(EID, Fname, Lname, MI)
Employee_SSN(SSN, EID)
Contact_Phone_info(EID, Phone)
Employee_Bdate(EID, Bdate)
Employee_JobTitle(EID,Job_Title[2])
Employee_Salary(EID, Salary)
Contact_Address_Info(EID,Street, City, Zip)
Employee_State(Zip, State)
Employee_Email(EID, Email)

This final result after normalization does not violate BCNF, because no non-primary key could determine a primary key.

3. Order_Details (Order_ID[20], Product_ID[12b], Quantity, UnitPrice)

This relationship shows that each Order ID corresponds to product ID. The combination of them determines order's quantity and unit price of the product. To normalize this relationship, consider from 1NF, 2NF, 3NF, and BCNF.

1NF: Order_Details is in 1NF because there is no multi-value attributes.

Order_Details (Order_ID[20], Product_ID[12b], Quantity, UnitPrice)

2NF: The relationship is not in 2NF since partial determination exists. Order_ID→ Quantity of orders, and Product_ID→ UnitPrice of the product. After normalizing it, we get:

Order_Details1 (Order_ID[20], Quantity)
Order_Details2 (Product_ID[12b], UnitPrice)

3NF: Above normalization we have qualifies for 3NF since no non-primary keys determines each other.

Order_Details1 (Order_ID[20], Quantity)
Order_Details2 (Product_ID[12b], UnitPrice)

This final result after normalization does not violate BCNF, because no non-primary key determines a primary key.

4. Product (Inventory_ID[12], Product_ID, Order_ID[20], Product_Type_ID[14],Date_Produced, Timeslot_ID[17], EID[1a])

This relationship records all indicators related to product, including how a product is related to order, inventory, timeslot, and employee who has produced this product.. To normalize this relationship, consider from 1NF, 2NF, 3NF, and BCNF.

1NF: Product is in 1NF because there is no multi-value attributes.

Product (Inventory_ID[12], Product_ID, Order_ID[20], Product_Type_ID[14],Date_Produced, Timeslot_ID[17], EID[1a])

2NF: No partial dependency in this relationship, so at this step, the normalized relations remain the same.

Product (Inventory_ID[12], Product_ID, Order_ID[20], Product_Type_ID[14],Date_Produced, Timeslot_ID[17], EID[1a])

3NF: Inventory_ID → Product_ID because each inventory ID corresponds to one specific product (product ID). Product_ID→ Product_Type_ID, each product_id also represents its product type. Both Product_ID and Product_Type_ID are not primary keys, but they determine each other. This violates 3NF rule. After normalization we could conclude:

Product (IInventory_ID[12], Product_ID,Order_ID[20],Date_Produced, Timeslot_ID[17], EID[1a])
Product_Product_Type(Product_ID, Product_Type_ID[14])

This final result after normalization does not violate BCNF, because no non-primary key determines a primary key.

5. Recipe_Nutrition_List (Recipe_ID, Nutrition_Name, Nutrition_ID)

1NF: This relationship lists all the nutritions in each recipe. Nutrition_ID is determined by both Recipe_ID and Nutrition_Name (ex: Calcium, VitaminC, etc). To normalize this relationship, consider from 1NF, 2NF, 3NF, and BCNF.

1NF: Product is in 1NF because there is no multi-value attributes.

Recipe_Nutrition_List (Recipe_ID, Nutrition_Name, Nutrition_ID)

2NF: No partial dependency in this relationship, so at this step, the normalized relations remain the same.

Recipe_Nutrition_List (Recipe_ID, Nutrition_Name, Nutrition_ID)

3NF: Above normalization we have qualifies for 3NF since no non-primary keys determines each other.

Recipe_Nutrition_List (Recipe_ID, Nutrition_Name, Nutrition_ID)

BCNF: In this case, Nutrition_ID would determine Nutrition_Name, but Nutrition_ID is not a superkey. So the relationship violates BCNF. RecipeID and NutritionID can combine to determine nutritional values in each recipe. Rewrite Recipe_Nutrition_List as:

NutritionName(Nutrition_ID, Nutrition_Name)

Recipe_Nutrition_Value(Recipe_ID, Nutrition_ID)

Discussion and Future Work

Through our work with Uptown Juice Company we learned a lot about the complexity of designing a database for a real company. Not only did we get to experience the level of detail that one must go into when thinking about how to properly map all the data-related needs of a company, but also we got to see how important data is when it comes to taking business decisions. Having a system that goes beyond simple information tracking can truly impact how a company operates. Even though we understand that our product is just a prototype, our objective has always been to positively influence our client's future by illustrating how the data they keep track of plays a key role in their day to day operations. Hopefully our collaboration with Uptown Juice Company will help them fully understand the potential that having a complex database system can unleash for their company.

We believe that understanding demand is an important part of a company's success. For our client, knowing where their orders and customers are coming from was an important part of their future expansion plans. Through our database and queries, we were able to take a closer look at how demand for Uptown Juice Co.'s products is distributed geographically. Since we were unable to acquire real data, we can not provide our client with a true recommendation of where to open up additional store locations. However, we have provided Uptown Juice Co. with the tools to understand how they can use data in order to help them decide where to place additional store locations if they choose to expand. As a recommendation to our client, we believe that they should seek to implement similar models to the ones provided in this report and use real data in order to help them craft future expansion plans. At this moment, the customer complaint table does not exist in Uptown's database, and for each product, not all the relevant information is obtained in the database either. So it is not possible to trace back to the production team or the suppliers. However, the company is very interested in this feature and sees the value of having this available because the contaminated product problem could be very critical in the food services. We do hope this idea could be used for them to minimize their damage. As for the supplier, in addition to increasing/decreasing trend, there might also be seasonal trend, that cannot be captured using the data that we generated. But this could also be an important part to work on in the future.

The data we used for the marketing analysis is what we made based on the products sold at the store. In the future, when more data from the client are available, we can build more accurate models for this system. Also we plan to connect our database with R, making it a real time simulator and a more convenient and less time-consuming app to meet the demand for expanded dataset from the client.

We have generated data based on reasonable assumptions, but is not accurate to what the Uptown obtains. Once we have the real data, the time series model will need to be adjusted, but the overall approach should remain the same. Our goal is to have this forecasted August demand on hand to compare with the real demand in August to test the model. This is another query that Uptown looks forward to obtain in their database since inventory planning is also a

huge concern for them. They think it would be much more efficient if they have this query available.

Employee scheduling is completely dependent on the forecasted demand. Once the database gets implemented and the forecasting model gets adjusted, the parameter values in the linear programming will need to be adjusted as well as some constraint values. But the same tool and code could be used for this purpose. The scheduling plan should be able to minimize Uptown's labor cost in order to maximize its profit.

IX. Appendix

Figure 1: Dollar Spending per Zip Code

Node	Zip Code	Demand
1	94601	482
2	94602	485
3	94603	474
4	94605	511
5	94606	512
6	94607	493
7	94608	973
8	94609	930
9	94610	493
10	94611	447
11	94612	451
12	94618	930
13	94619	473
14	94621	497
15	94702	460
16	94703	475
17	94704	480
18	94705	479
19	94706	459
20	94707	504
21	94708	466
22	94709	483
23	94710	484
24	94720	521

Figure 2: Distance Between Nodes (in Miles)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	1.6	3.7	3.2	1.8	3.8	5.4	4.5	2.7	3.4	3.3	4.6	1.6	2.3	6.9	6.5	6.5	5.7	8.7	8.8	8.2	7.3	7.4	6.7
2	1.6	0	4.7	3.8	1.9	3.8	4.7	3.6	2.1	2	3.1	3.4	1.5	3.5	5.9	5.4	5.3	4.4	7.7	7.6	6.9	6.1	6.5	5.3
3	3.7	4.7	0	1.6	5.5	7.4	9	8.1	6.4	6.7	6.9	8	3.4	1.4	10.5	10	10	9.1	12.3	12.3	11.6	10.8	11.1	10
4	3.2	3.8	1.6	0	4.9	7	8.3	7.3	5.6	5.7	6.4	7	2.2	1.6	9.6	9.1	8.9	8	11.3	11.1	10.4	9.7	10.2	8.9
5	1.8	1.9	5.5	4.9	0	2.1	3.6	2.9	1.1	2.3	1.5	3.3	3	4.1	5.2	4.9	5	4.3	7.1	7.3	6.8	5.8	5.7	5.3
6	3.8	3.8	7.4	7	2.1	0	2.2	2.2	1.9	3.2	0.7	3.3	5.1	6.1	4.1	4	4.4	4	5.9	6.4	6.2	5.1	4.4	5
7	5.4	4.7	9	8.3	3.6	2.2	0	1.2	2.7	4	2.2	3	7.4	8.2	2	2.4	3.7	3.3	3.8	4.5	4.8	3.5	2.7	3.4
8	4.5	3.6	8.1	7.3	2.9	2.2	1.2	0	2.1	2.4	1.8	1.5	6	7.5	2.4	2.2	2.8	2.2	4.2	4.5	4.5	3.1	3.5	2.8
9	2.7	2.1	6.4	5.6	1.1	1.9	2.7	2.1	0	2	1.6	2.3	4.1	5.5	4.5	4.2	4.2	3.5	6.2	6.5	6.3	4.9	5.6	4.4
10	3.4	2	6.7	5.7	2.3	3.2	4	2.4	2	0	3.2	1.2	4.3	6.7	4.1	3.7	2.8	2.2	5.5	5.4	5	3.9	5.4	3.3
11	3.3	3.1	6.9	6.4	1.5	0.7	2.2	1.8	1.6	3.2	0	2.9	5.5	6	3.9	3.9	4.6	3.9	5.8	6.3	6.3	4.2	4.8	4.6
12	4.6	3.4	8	7	3.3	3.3	3	1.5	2.3	1.2	2.9	0	5.4	7.7	2.9	2.4	1.9	1.1	4.4	4.4	4	2.8	4.2	2.2
13	1.6	1.5	3.4	2.2	3	5.1	7.4	6	4.1	4.3	5.5	5.4	0	4.2	8.3	7.9	7.1	7.4	9.8	9.7	9.2	8.2	9.5	7.6
14	2.3	3.5	1.4	1.6	4.1	6.1	8.2	7.5	5.5	6.7	6	7.7	4.2	0	9.9	9.7	9.7	8.9	11.7	12	11.7	10.4	10.8	9.9
15	6.9	5.9	10.5	9.6	5.2	4.1	2	2.4	4.5	4.1	3.9	2.9	8.3	9.9	0	0.6	2.3	2.4	1.8	2.5	2.8	1.6	1.2	1.9
16	6.5	5.4	10	9.1	4.9	4	2.4	2.2	4.2	3.7	3.9	2.4	7.9	9.7	0.6	0	1.7	1.8	2	2.3	2.5	1.1	1.8	1.2
17	6.5	5.3	10	8.9	5	4.4	3.7	2.8	4.2	2.8	4.6	1.9	7.1	9.7	2.3	1.7	0	0.8	3	2.6	2.1	1.2	3.4	0.6
18	5.7	4.4	9.1	8	4.3	4	3.3	2.2	3.5	2.2	3.9	1.1	7.4	8.9	2.4	1.8	0.8	0	3.5	3.3	2.9	1.8	3.6	1.1
19	8.7	7.7	12.3	11.3	7.1	5.9	3.8	4.2	6.2	5.5	5.8	4.4	9.8	11.7	1.8	2	3	3.5	0	1.1	1.9	1.8	1.4	2.4
20	8.8	7.6	12.3	11.1	7.3	6.4	4.5	4.5	6.5	5.4	6.3	4.4	9.7	12	2.5	2.3	2.6	3.3	1.1	0	0.9	1.6	2.5	2.2
21	8.2	6.9	11.6	10.4	6.8	6.2	4.8	4.5	6.3	5	6.3	4	9.2	11.7	2.8	2.5	2.1	2.9	1.9	0.9	0	1.4	3.2	1.8
22	7.3	6.1	10.8	9.7	5.8	5.1	3.5	3.1	4.9	3.9	4.2	2.8	8.2	10.4	1.6	1.1	1.2	1.8	1.8	1.6	1.4	0	2.4	0.66
23	7.4	6.5	11.1	10.2	5.7	4.4	2.7	3.5	5.6	5.4	4.8	4.2	9.5	10.8	1.2	1.8	3.4	3.6	1.4	2.5	3.2	2.4	0	2.9
24	6.7	5.3	10	8.9	5.3	5	3.4	2.8	4.4	3.3	4.6	2.2	7.6	9.9	1.9	1.2	0.6	1.1	2.4	2.2	1.8	0.7	2.9	0

Query 3: Implementations in R

Data from SQL (first 20 records)

Product_ID	EventID	type	duration	unit_price	unit_cost	discount
32581	1	1	12	8.99	3.91	0.1
33536	1	1	11	8.99	4.27	0.1
47530	1	1	10	8.99	3.55	0.1
32766	1	1	13	8.99	4.26	0.1
38943	1	1	9	8.99	4.11	0.1
35725	1	1	12	8.99	3.03	0.1
37644	1	1	12	8.99	3.28	0.1
43754	1	1	11	8.99	3.86	0.1
46486	1	1	10	8.99	4.32	0.1
34205	1	1	12	8.99	3.58	0.1
39994	1	1	11	8.99	3.43	0.1
35871	1	1	11	8.99	3.40	0.1
47641	1	1	11	8.99	4.16	0.1
41796	1	1	10	8.99	4.02	0.1
46080	1	1	13	8.99	3.61	0.1
32543	2	1	10	8.99	4.11	0.1
46169	2	1	11	8.99	3.37	0.1
35604	2	1	11	8.99	3.25	0.1
36572	2	1	14	8.99	4.66	0.1
35737	2	1	12	8.99	4.28	0.1

marketing_script.R

```
data = read.csv("datafromSQL.csv")
data$type = as.factor(data$type)

library(ggplot2)
ggplot(data, aes(factor(type), duration)) + geom_boxplot(aes(fill = factor(type)))

m = lm(duration~type, data = data)
summary(m)
pred = predict(m, newdata = data.frame(type=factor(1:8)))
res = data.frame(type=1:8, duration=round(pred))

lm_coef = function(i){
  temp = subset(data, (EventID==1 | EventID==2 | EventID==3 |
                         EventID==4 |EventID==5 | EventID==6) & type==i)
  t = sapply(split(temp,as.factor(temp$EventID)), nrow)
  d = sapply(split(temp,as.factor(temp$EventID)), function(x)unique(x$discount))
  coef(lm(t~d))
}

wktable = as.data.frame(matrix(c(lm_coef(1),lm_coef(2),lm_coef(3),lm_coef(4),
                                 lm_coef(5),lm_coef(6),lm_coef(7),lm_coef(8)),
                               nrow=8,byrow = TRUE))
colnames(wktable) = c("b","a")
ctable = cbind(res, wktable)
ctable$b = round(ctable$b, digits = 2)
ctable$a = round(ctable$a, digits = 2)
```

server.R

```
library(shiny)
source("marketing_script.R")
shinyServer(function(input, output) {
  result1 = reactive({
    t = c("Almond Milk", "Soy Milk", "Apple Juice","Orange Juice",
          "Veggie Juice", "Detox Juice","Lemonade","Protein Juice")
    i = which(t%in%input$type)
    ct = ctable[i,]
    du = ct[2]
    du
  })
  result2 = reactive({
    t = c("Almond Milk", "Soy Milk", "Apple Juice","Orange Juice",
          "Veggie Juice", "Detox Juice","Lemonade","Protein Juice")
    i = which(t%in%input$type)
    ct = ctable[i,]
    validate(
      need(input$cost != "", "Please enter a cost"),
      need(input$cost >= 0, "A valid cost should >= 0"),
      need(input$price >= 0, "A valid sale price should >= 0"),
      need(input$cost <= input$price, "A valid cost should <= sale price")
    )
    dis = round((0.5 - ct[3]/(2*ct[4]) - input$cost/(2*(input$price)))*100,2)
    dis
  })
  output$text1 <- renderText({
    paste("Sale Price: $", input$price, sep="")
  })
  output$text2 <- renderText({
    paste("Cost: $", input$cost, sep="")
  })
  output$dis <- renderText({
    discount = result2()
    if (discount != Inf){
      paste("Recommended Discount for ", input$type, ": ", discount,"% off", sep="")
    }
    else{
      paste("No promotion needed for ",input$type, sep = "")
    }
  })
  output$du <- renderText({
    t = c("Almond Milk", "Soy Milk", "Apple Juice","Orange Juice",
          "Veggie Juice", "Detox Juice","Lemonade","Protein Juice")
    i = which(t%in%input$type)
    ct = ctable[i,]
    if (abs(ct[3]/ct[4]) != Inf){
      paste("Recommended Promo Period: ", result1()," days", sep="")
    }
  })
})
```

ui.R

```
library(shiny)
shinyUI(fluidPage(
  titlePanel("Promotion Generator"),
  sidebarLayout(
    sidebarPanel(
      helpText("Provide promotion suggestion based on 2016 sales information
               for Uptown Juice Company"),
      img(src = "logo.png", height = 120, width = 250),
      selectInput("type",
                  label = "Choose a product",
                  choices = list("Almond Milk", "Soy Milk", "Apple Juice",
                                 "Orange Juice", "Veggie Juice", "Detox Juice",
                                 "Lemonade","Protein Juice"),
                  selected = "Almond Milk"),
      sliderInput(inputId = "price",
                  label = "Sale Price",
                  min = 0,
                  max = 50,
                  value = 2.99,
                  step = 0.01),
      numericInput(inputId = "cost",
                   label = "Unit Cost",
                   min = 0,
                   max = 10,
                   value = 0)
    ),
    mainPanel(
      textOutput("text1"),
      textOutput("text2"),
      textOutput("dis"),
      textOutput("du"),
      tags$head(tags$style("#dis{color: green;
                                 font-size: 20px;
                           font-style: bold;
                           }", "#du{color: blue;
                           font-size: 20px;
                          font-style: bold;
                          }","#text1{color: purple;
                           font-size: 16px;
                           font-style: bold;
                           }","#text2{color: orange;
                           font-size: 16px;
                           font-style: bold;
                           }"
                           )
      ),
      br(),
      br(),
      br(),
      img(src="orig.jpg",height = 180, width = 350)
    )
))
```

Updated Normalized Relation Design

1. Employee_Name(EID, Fname, Lname, MI)
   Employee_SSN(SSN, EID)
   Contact_Phone_info(EID, Phone)
   Employee_Bdate(EID, Bdate)
   Employee_JobTitle(EID,Job_Title2)
   Employee_Salary(EID, Salary)
   Contact_Address_Info(EID,Street, City, Zip)
   Employee_State(Zip, State)
   Employee_Email(EID, Email)

1a. Chopper (ChooperID, EID[1a])

Chopper_Speed(ChopperID, Speed)

1b. Juice_Specialist (JSID, EID[1a])
Juice_Specialist_Experience (JSID, rating, exprience)

1c. Packer (PackerID,EID[1a])
Packer_Efficiency(PackerID, Efficiency)

1d Part_Time_Employee (PTEID, EID[1a], Hourly_Wage)
Part_Time_Employee_Availability(EID, Availability)
Part_Time_Employee_Wage(PTEID, Hourly_Wage)

1e Marketing_Employee(MarketingID,EID[1a])
Marketing_Employee_Flyer(MarketingID, Flyer_Quantity)

1f Procurement(ProcurementID, EID[1a])
Procurement_Business_Card(ProcurementID, Business_Card_Quantity)

1g Delivery_Employee(DEID, EID[1a], DriversLicense)
Delivery_Employee_DL(DEID, DriverLicense)

1h Sale_Employee(SaleEID, EID[1a])
Salesman_Commission(SaleEID, Commission)

2. Job (Job_Title, Job_Responsibilities)
Job_Requirement (Job_Title, Job_Requirement)
Job_Title_Description(Job_Title, Job_Description)

3. Non-Retail_Customer (CID, Fname, Lname, Emails, Phone)
Non-Retail_Customer(CID, Email)

3a. Delivery_Customer(CID3, Street, City, Zip)
Customer_State(Zip, State)
3b. Rewards_Customer (CID[3], Bdate)

4. Rewards_Customer_Coupon(CID[3], Promotion_ID[24a])

5. Delivery(DeliveryID, Delivery_Date, Order_ID, CID[3])
Order_DollarAmount(Order_ID, Dollar_Amount)
5a. UPtown_Delivery (Delivery_Date[5,] Delivery_ID[5], Delivered_By_Employee[1a])
5b. Third_Party_Delivery (Delivery_Date[5,] Delivery_ID[5], Delivered_By)
(dom(Delivered_By) ={Caviar, UberEats})

6. Recipe(RecipeID, Ingredient_ID[12a], Quantity)
RecipeNutritonList(RecipeID, Nutrion_Name, NutrionID)

7. Label (Label_ID,Attachment1, Attachment2)

8. Nutrition_Fact (Nutrition_ID, Nutrition_Name, Nutrition_Value)

9. Supplier (SID, Company_Name, Contact_Person, Discount)
Supplier_Email(SID, Email)
Contact_Person_Phone(SID, Phone)
Supplier_Address(SID, Street, City, Zip)
Supplier_State(Zip, State)

10. Type_of_Fixed_Asset(Asset_Code (1 to 12),Type, Quantity, Purchase_ID)
dom(Type)= {Office, Kitchen, Vehicle, }
11. Individual_Fixed_Asset(Asset_ID, Asset_Code[10])
12. Inventory (Inventory_ID, Quantity, Name)
12a. Ingredient (Inventory_ID[12], Purchase_ID[15])
IngredientInventory(Inventory_ID, Ingredient_ID)
IngredientTypeIngredient(Ingredient_ID, Ingredient_Type_ID[13])
Ingredient_Quantity(Ingredient_ID, Quantity)
12b. Product (Inventory_ID[12], Product_ID, Order_ID[20],Date_Produced, Timeslot_ID[17], EID[1a])
Product_Product_Type(Product_ID, Product_Type_ID[14])
13. Ingredient_Type(Ingredient_Type_ID, Name, Seasonalities)
Ingredient_Seasonality(Ingredient_Type_ID, Seasonality)
dom(Seasonality)={spring,summer,fall,winter}
14. Product_Type(Product_Type_ID, Product_Name, RecipeID, UnitPrice)
15. Purchase (Purchase_ID, EID[1a], SID[9], Timeslot_ID[17], Date_Purchased, Date_Received, Date_Charged, Quantity, Type) (dom(Type)={Ingredient, FixedAsset})
16. Product_Detail (Ingredient_ID, Product_ID)
17. Timeslot (Timeslot_ID, Expiration_Date)
17a. Timeslot_For_Purchase (Timeslot_ID[17])
17b. Timeslot_For_Product (Timeslot_ID[17,] Shelf_Life)
18. Shift_Schedule_For_Part_Time_Employee (Shift_ID, Date, Shift_Start, Shift_End, Hour)
19. Shift_Worker(Shift_ID, EID[1a])
20. Order (Order_ID, EID[1a], CID[3], Date_Placed, Time_Placed, Date_Posted, Time)
Payment_Method(Order_ID, Payment_Method)
Order_Placed_Date(Date_Placed, Day)
(dom(Time)={Morning, Afternoon, Night},
dom(Day)={Monday,Tuesday,Wednesday,Thursday,Friday, Saturday, Sunday}
21. Order_Details1 (Order_ID[20], Quantity)
Order_Details2 (Product_ID[12b], UnitPrice)
22. Payment(Payment_ID, DollarAmount)
22a. Purchase_Payment (Payment_ID,Purchase_ID[15])
22b. Utilities_Payment(Payment_ID, Utility_Type)
23. Revenue(Revenue_ID, DollarAmount)
23a. Order_Revenue (Revenue_ID, Order_ID[20])
23b. Promotion_Revenue (Revenue_ID, Event_ID[24])
24. Event (Event_ID, Organizer_EID[2], Event_Date)
24a) Promotion_Event(Event_ID[24], Product_ID[12b], Discount, Quantity_Sold, Duration, Revenue_ID)
24aa. Customer_Promotion (Event_ID[24])
24ab. Product Promotion (Event_ID[24], Timeslot_ID[17b])

24ac. Seasonality_Promotion (Event_ID[24],Seasonality[13])
24b) Donation_Event (Event_ID[24])
25. Customer_Complaint (Complaint_ID, Order_ID[20])