Online Auction Database Project

IEOR 115 || Professor Ken Goldberg || Fall 2014

Shirley Bao, Jessica Huang, Jie Li, Cindy Mo, Jeremy Wan,
Yijian Wu, Kangting Yu, Aurona Zhang

Sellegit Background

Online Marketplace for Safe Transactions near College Campuses

Downtown Berkeley Startup

Conventional Auction Business Model

Founded by Berkeley Alumni – Go Bears!!

Meet the Founders

Peter Chen Jeff Zhang George Zhang Rocky Duan Peter Qian

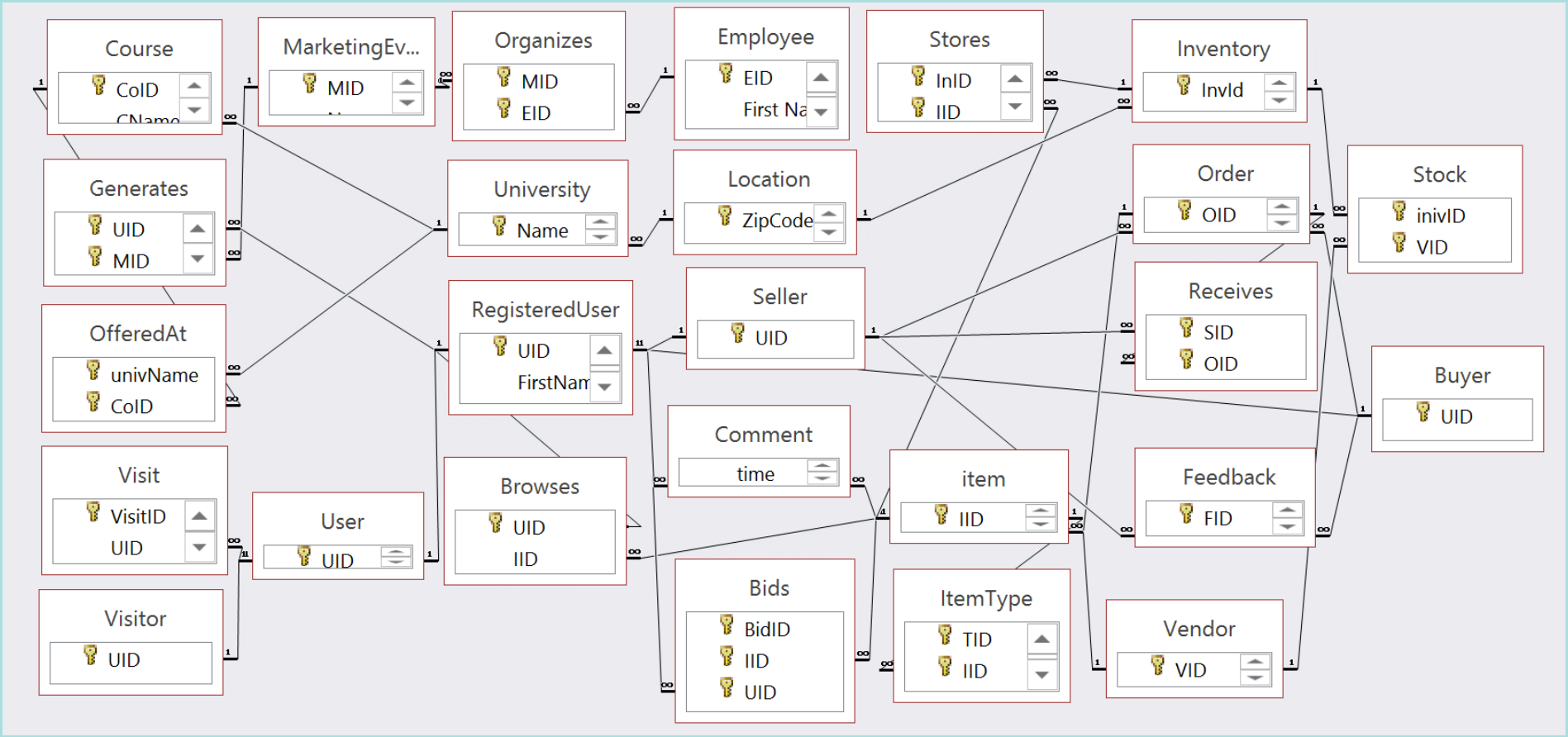
Database Schema

1. Visit(VisitID, User[2], IPaddress, loginTime, duration, numClick)
2. User(UID, frequency, device)
   a. Visitor(UID[2], ...)
   b. RegisteredUser(UID[2], Fname, Lname, email, phoneNumber, sellegitCredits, referralCode, password, university[6])
      i. Buyer(UID[2], ...)
      ii. Seller(UID[2], ...)
3. Employee(EID, Fname, Lname, email, phoneNumber, title, DOB)
4. Item(IID, listPrice, mostRecentBidPrice, status, numBids, description, datePosted, minPrice, condition, vendor[8], location[9], order[10], seller[2b(ii)])
5. ItemType(TID, item[4], typeName, makeYear, material, brand, madeIn, course[7], faceValue, detailedDescription)
6. University(univName, startDate, endDate, size, location[9])
7. Course(CoID, cName, cNum, semester, year, location[9])
8. Vendor(VID, vName, industry, size, location[9])
9. Location(zipCode, city, state, country)

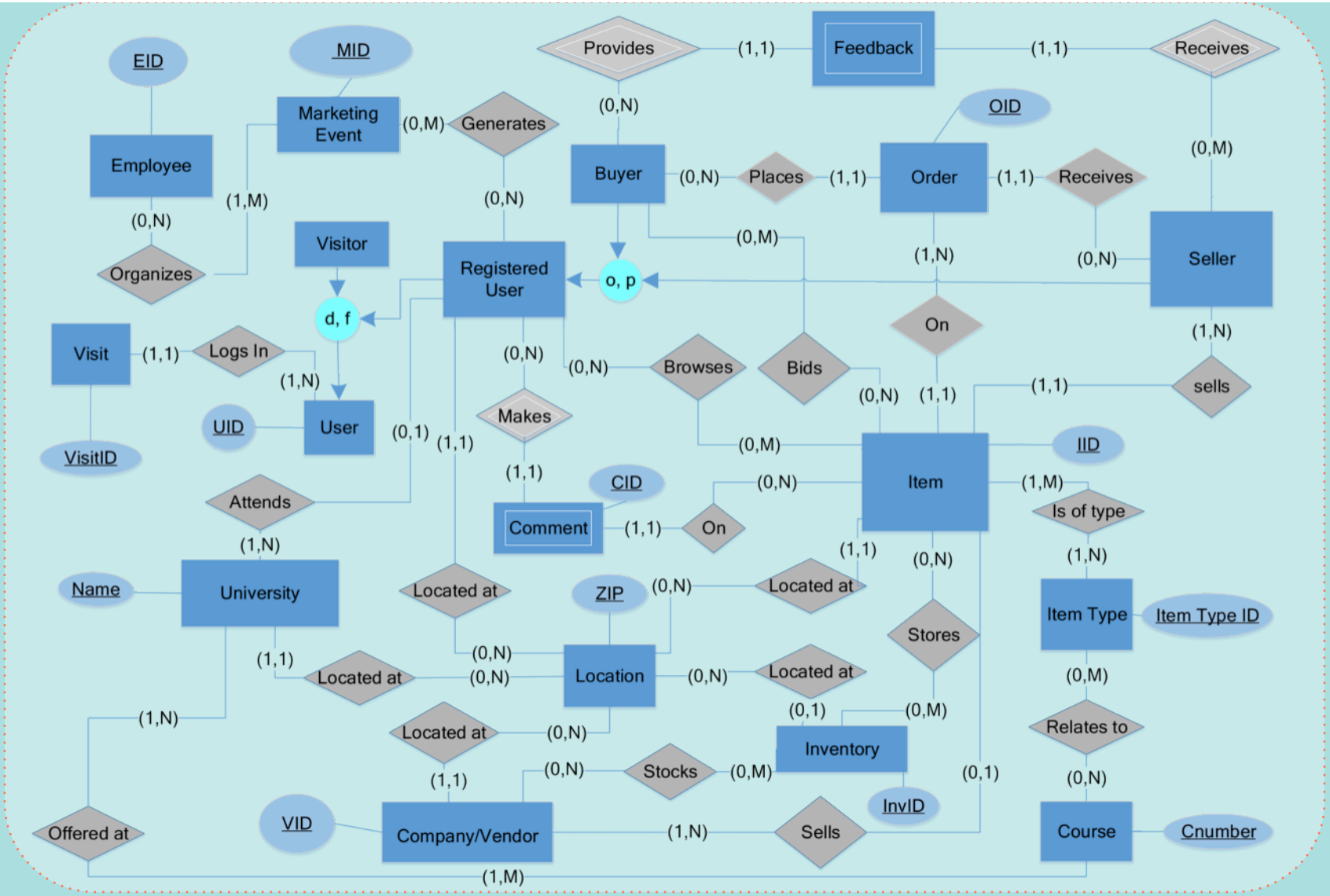
Database Schema

10. Order(OID, time, paymentType, paymentAmt, pickupTime, location[9], item[4], buyer[2b(i)], seller[2b(ii)])
11. MarketingEvent(MID, name, date, platform, type)
12. Comment(CID, time, content, item[4], user[2b])
13. Feedback(FID, time, content, rating, buyer[2b(i)], seller[2b(ii)])
14. Inventory(InvID, location[9], stockDate, size, clearBy)
15. Generates(user[2b,] marketingEvent[11])
16. Organizes(employee[3,] marketingEvent[11])
17. Bids(buyer[2b(i)], item[4])
18. Receives(seller[2b(i)], order[10])
19. Browses(user[2b], item[4])
20. IsOfType(item[4], type[5])
21. Stores(inventory[14], item[4])
22. Stocks(vendor[8,] inventory[14])
23. RelatesTo(textbook[5], course[7])
24. OfferedAt(university[6], course[7])

Relationships in MS Access



EER Diagram


EID
MID
Provides
(1,1)
Feedback
(1,1)
Receives
Marketing Event
(0,M)
Generates
(0,N)
OID
Employee
(1,M)
(0,N)
Buyer
(0,N)
Places
(1,1)
Order
(1,1)
Receives
(0,M)
(0,N)
Visitor
(0,M)
(1,N)
(0,N)
Seller
Organizes
Registered User
o, p
d, f
On
(1,N)
Visit
(1,1)
Logs In
(1,N)
(0,N)
(0,N)
Browses
Bids
(0,N)
(1,1)
(1,1)
sells
UID
User
(0,1)
(1,1)
Makes
(0,M)
IID
VisitID
(1,1)
CID
(0,N)
Item
(1,M)
Attends
Comment
(1,1)
On
Is of type
(1,N)
(1,N)
(1,1)
(0,N)
Name
University
Located at
ZIP
(0,N)
Located at
Stores
Item Type
Item Type ID
(0,N)
(1,1)
Located at
(0,N)
Location
(0,N)
Located at
(0,M)
(1,N)
(0,1)
(0,M)
Relates to
Located at
(0,N)
Inventory
(0,N)
Stocks
(0,M)
(0,1)
(1,1)
(0,N)
InvID
Offered at
VID
Company/Vendor
(1,N)
Sells
Course
Cnumber
(1,M)

QUERIES

Query 1: Listing Price Recommendation

At what price should a seller list his or her item to ensure a quick sell?

Determine face value of item

+

Determine item's category

↓

Apply percentage to current item

+

Calculate past items' percentage change

Query 1: Listing Price Recommendation

Business Justification:

- ✓ Helps give users **peace of mind** when listing and pricing an item for auction
- ✓ Ensures **increased probability of selling** the item based on previous sales of similar type
- ✓ Provides **flexibility to the seller** to choose an appropriate listing price based on our recommendation

Query 1: Listing Price Recommendation

Implementation Process:

SQL

Extracts relevant data from the database

SAS

Run statistical analysis on past pricing trends

MS Access

Displays the appropriate results based on the SQL queries

Query 1: Listing Price Recommendation

SQL Code:

```
SELECT Order.OID, Order.original_price, Order.final_price, Order.IID AS Order_IID,
        ItemType.IID AS ItemType_IID, ItemType.TypeName,
        Round(((Order.original_price-Order.final_price)/Order.original_price),4) AS
        percentage_change
FROM [Order] INNER JOIN ItemType ON Order.IID = ItemType.IID
ORDER BY ItemType.TypeName DESC, Round(((Order.original_price-Order.final_price)/
            Order.original_price),4) DESC;
```

Query 1: Listing Price Recommendation

Sample Output:

Order Query

OID	original_price	final_price	Order_IID	ItemType_IID	TypeName	percentage_c
6	199	90.01	6	6	game	0.5477
24	99.99	62	24	24	game	0.3799
32	5	4.94	32	32	game	0.012
45	10	0.25	45	45	furniture	0.975
31	15	1.05	31	31	furniture	0.93
14	28	5.01	14	14	furniture	0.8211
42	80	15	42	42	furniture	0.8125
41	80	15	41	41	furniture	0.8125
48	238	45	48	48	furniture	0.8109
17	40	10	17	17	furniture	0.75
39	60	16	39	39	furniture	0.7333
38	69.98	20.02	38	38	furniture	0.7139
46	50	20	46	46	furniture	0.6
10	69	29.93	10	10	furniture	0.5662
11	69	29.93	11	11	furniture	0.5662
23	149	70	23	23	furniture	0.5302
1	45	21.29	1	1	furniture	0.5269

Query 1: Listing Price Recommendation

SAS Code:

```
libname exdat excel 'C:\Users\student\Desktop\
    query1full.xlsx';
data full; set exdat.'Order Query$'n;
proc import out = full
    datafile = 'C:\Users\student\Desktop\query1full.xlsx'
    dbms = excel replace;
    range = 'Order Query$'n;
run;

proc means data = full mean median q1 q3 std;
class typeName;
var percentage_change;
title ' ';

data furniture; set full;
if typeName = 'furniture';
run;

proc univariate data = furniture noprint;
   var percentage_change;
   output out=percnt pctlpts = 5 50 95 pctlpre = P;
```

```
title "5 50 95 percentile of percentage change for
   funitures";
proc print data = percnt;

ods graphics off;
symbol v=plus;
title 'Normal Quantile-Quantile Plot for Percentage
Change';

proc capability data = furniture noprint;
  spec lsl = 0.3 usl = 0.8;
  qqplot percentage_change;
run;

proc univariate data = furniture;
histogram percentage_change / normal
midpoints = (0.1 to 0.9 by 0.1);
title "Histogram of Percentage Change(Furnitures)";
run;
```

Query 1: Listing Price Recommendation

SAS Output:

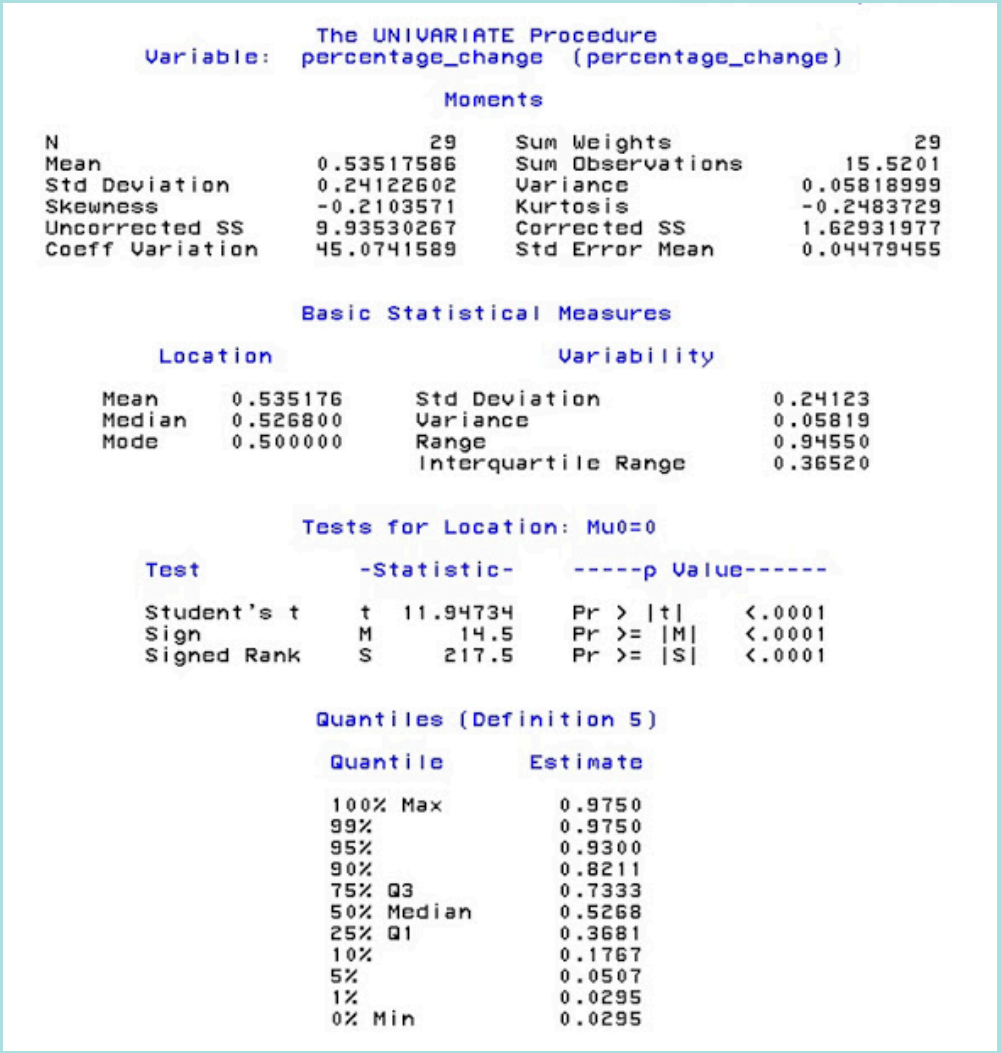
The MEANS Procedure

Analysis Variable : percentage_change percentage_change

TypeName	N Obs	Mean	Median	Lower Quartile	Upper Quartile	Std Dev
book	3	0.3737333	0.3743000	0.2500000	0.4969000	0.1234510
clothing	4	0.4131000	0.4701000	0.1915000	0.6347000	0.3019002
electronic	11	0.4975000	0.5000000	0.2744000	0.6000000	0.2400388
furniture	29	0.5351759	0.5268000	0.3681000	0.7333000	0.2412260
game	3	0.3132000	0.3799000	0.0120000	0.5477000	0.2740078

Query 1: Listing Price Recommendation

SAS Output:



The UNIVARIATE Procedure
Variable: percentage_change (percentage_change)

Moments

N	29	Sum Weights	29
Mean	0.53517586	Sum Observations	15.5201
Std Deviation	0.24122602	Variance	0.05818999
Skewness	-0.2103571	Kurtosis	-0.2483729
Uncorrected SS	9.93530267	Corrected SS	1.62931977
Coeff Variation	45.0741589	Std Error Mean	0.04479455

Basic Statistical Measures

Location		Variability	
Mean	0.535176	Std Deviation	0.24123
Median	0.526800	Variance	0.05819
Mode	0.500000	Range	0.94550
		Interquartile Range	0.36520

Tests for Location: Mu0=0

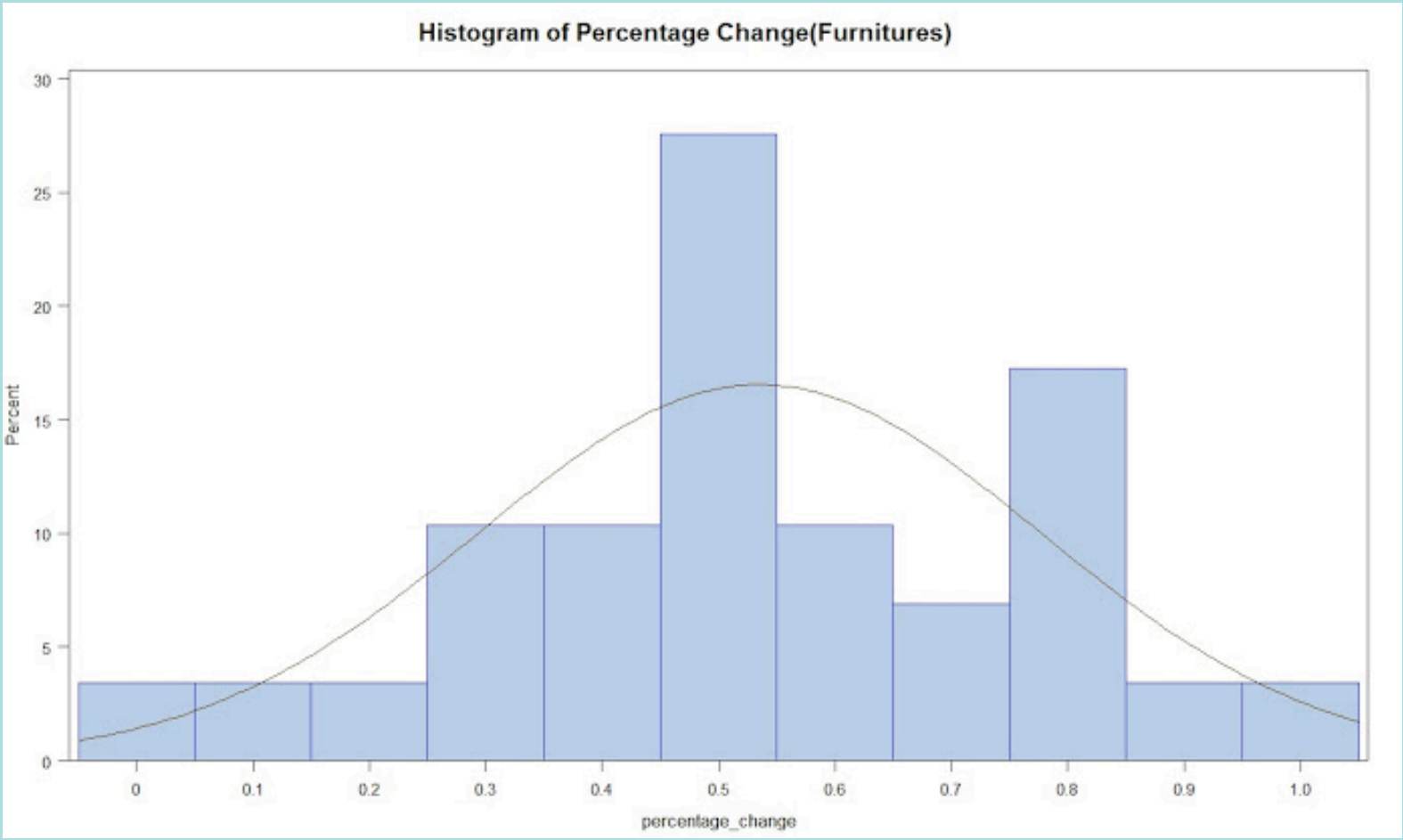
Test		-Statistic-	-----p Value------	
Student's t	t	11.94734	Pr > |t|	<.0001
Sign	M	14.5	Pr >= |M|	<.0001
Signed Rank	S	217.5	Pr >= |S|	<.0001

Quantiles (Definition 5)

Quantile	Estimate
100% Max	0.9750
99%	0.9750
95%	0.9300
90%	0.8211
75% Q3	0.7333
50% Median	0.5268
25% Q1	0.3681
10%	0.1767
5%	0.0507
1%	0.0295
0% Min	0.0295

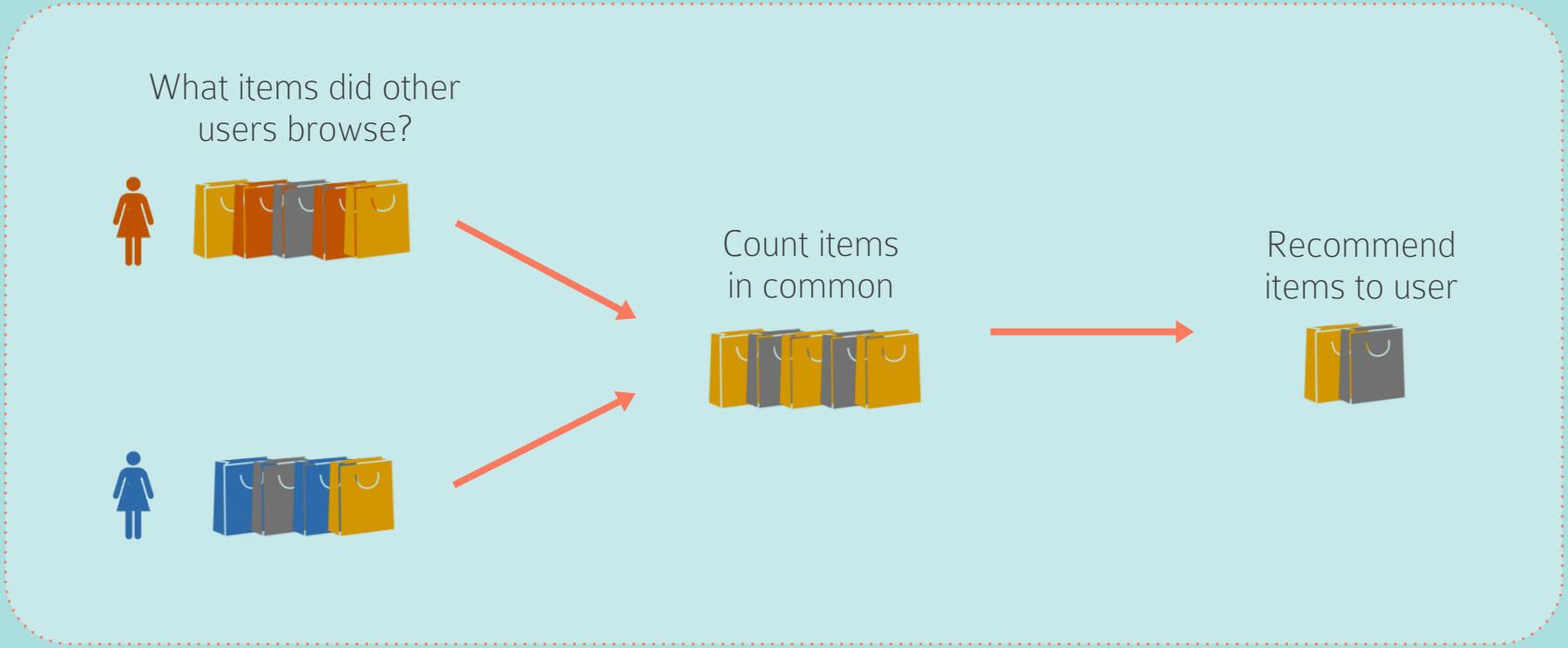
Query 1: Listing Price Recommendation

SAS Output:



Query 2: Item Recommendation

What other listed items would a Sellegit user be interested in purchasing?



Query 2: Item Recommendation

Business Justification:

- ✓ Extends the time a user stays on the website by providing many **relevant and eye-catching listings**
- ✓ **Increases the turnover rate** of listed items which provides incentive for sellers to use site
- ✓ Promotes **user loyalty** by automatically displaying other wanted items before leaving the website

Query 2: Item Recommendation

Implementation Process:

SQL

Extracts browsing and item data from the database

MS Access

Displays the appropriate results based on the SQL queries

Query 2: Item Recommendation

SQL Code:

```
SELECT item.IID, item.Name AS SuggestedItem
FROM item, (SELECT top 5 B.IID, COUNT(B.IID) AS C
            FROM Browses B, (SELECT B.UID FROM Browses B WHERE B.IID = 1) AS P
            WHERE B.UID = P.UID
            GROUP BY B.IID
            ORDER BY count(B.IID) DESC)  AS itemIID
WHERE (((item.IID)=[itemIID].[IID]));
```

Sample Output:

IID	SuggestedItem
1	IKEA single studying desk, black, wood
13	IKEA MALM 6-drawer Chest
12	21 forever girls shoes
34	IKEA Sultan Holmsta Full Size Spring Mattress, deliverable
37	Walmart single sofa, clearn

Query 2: Item Recommendation

Sample Output:

IID	SuggestedItem
5	IKEA single bed frame, deliverable
45	Storage
37	Walmart single sofa, clearn
12	21 forever girls shoes
33	Free Queen Size Mattress Giveaway
22	IKEA full size mattress, extra firm, 22' thick

IID	SuggestedItem
9	Walmart plastic chiar
4	IKEA wood chair,
45	Storage
23	Walmart full size mattress, extra firm, 18'thick, clearn
50	Canon Rebel T3i DSLR Camera

Query 2: Item Recommendation

USED **Aztec Print Cross Body Bag**

Original Price: $22.00

Watch Share

$5.00

Make Offer (0)

Requirement

Jessica Kim

DESCRIPTION

Item Specifics

Bought at Target

Recommendations

People are also viewing:

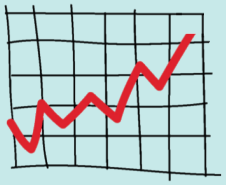
Say something

POST

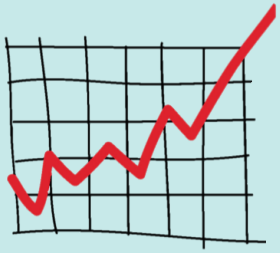
Query 3: Marketing Effort Optimization

When is the best time during the semester to deploy a Marketing Campaign for Sellegit?

Fit a time series model to site traffic data



Predict future site activity using model



Select appropriate phases for potential Marketing Campaigns

Query 3: Marketing Effort Optimization

Business Justification:

- ✓ Utilize web traffic logs to **model user activity**
- ✓ Analyze the **effectiveness of past marketing campaigns** using the established model
- ✓ Aid in **strategically timing their marketing efforts** to minimize wasteful campaigns

Query 3: Marketing Effort Optimization

Implementation Process:

SQL

Extracts web traffic data from the database

MS Access

Displays the web traffic results based on the SQL queries

R

Fit a time series model to web traffic data

Query 3: Marketing Effort Optimization

SQL Code:

```
SELECT Visit.LoginDate, Count(Visit.VisitID) AS Number_of_visitor_per_day
FROM Visit
GROUP BY LoginDate;
```

Sample Output:

LoginDate	Number_of_visitor_per_day
Fri Apr 25 2014	443
Sat Apr 26 2014	350
Thu Apr 24 2014	207

Query 3: Marketing Effort Optimization

R Code:

Load data

```
sellegit = read.csv("~/Sellegit_timeseries.csv", header = FALSE)
y = sellegit[,2]
```

Plot the original data

```
plot(y,type = 'l', col = 'red', xlab = 'month', ylab = 'number_of_visitors_per_day', xaxt = 'n')
axis(1, at = c(7,40,64,94,124,154,184,214,244,264,294,324), label = c("Nov", "Dec", "Jan", "Feb",
"March", "April", "May", "Jun", "Jul", "Aug", "Sep", "Oct"))
title("Sellegit Number of Visitors Over Months (Raw Data)")
```

Remove the fluctuation by taking 5 Day Moving Average

```
f10 = rep(1/5, 5)
y_lag=filter(y, f10, sides = 1)
plot(y_lag, col = "blue", xlab = 'month', ylab = 'number_of_visitors_per_day', xaxt='n')
axis(1,at=c(7,40,64,94,124,154,184,214,244,264,294,324), label = c("Nov", "Dec", "Jan", "Feb",
"March", "April", "May", "Jun", "Jul", "Aug", "Sep", "Oct"))
title("Sellegit Number of Visitors Over Months (5 Day Moving Average)")
```

Fit 2 half-semester data into seasonal AR(1) model

```
Spring_first_half = y_lag[74:145]
Fall_first_half = y_lag[294:365]
model_series_1 = c(Spring_first_half, Fall_first_half)
tsdisplay(model_series_1)
mod1 = sarima(model_series_1,0,1,0,1,0,0,72)
```

Query 3: Marketing Effort Optimization

R Output:

Seasonal AR(1) time series model:

$$\hat{Y}_t = \varphi_0 + \varphi_1 \cdot Y_{t-a} + e$$

Coefficients:

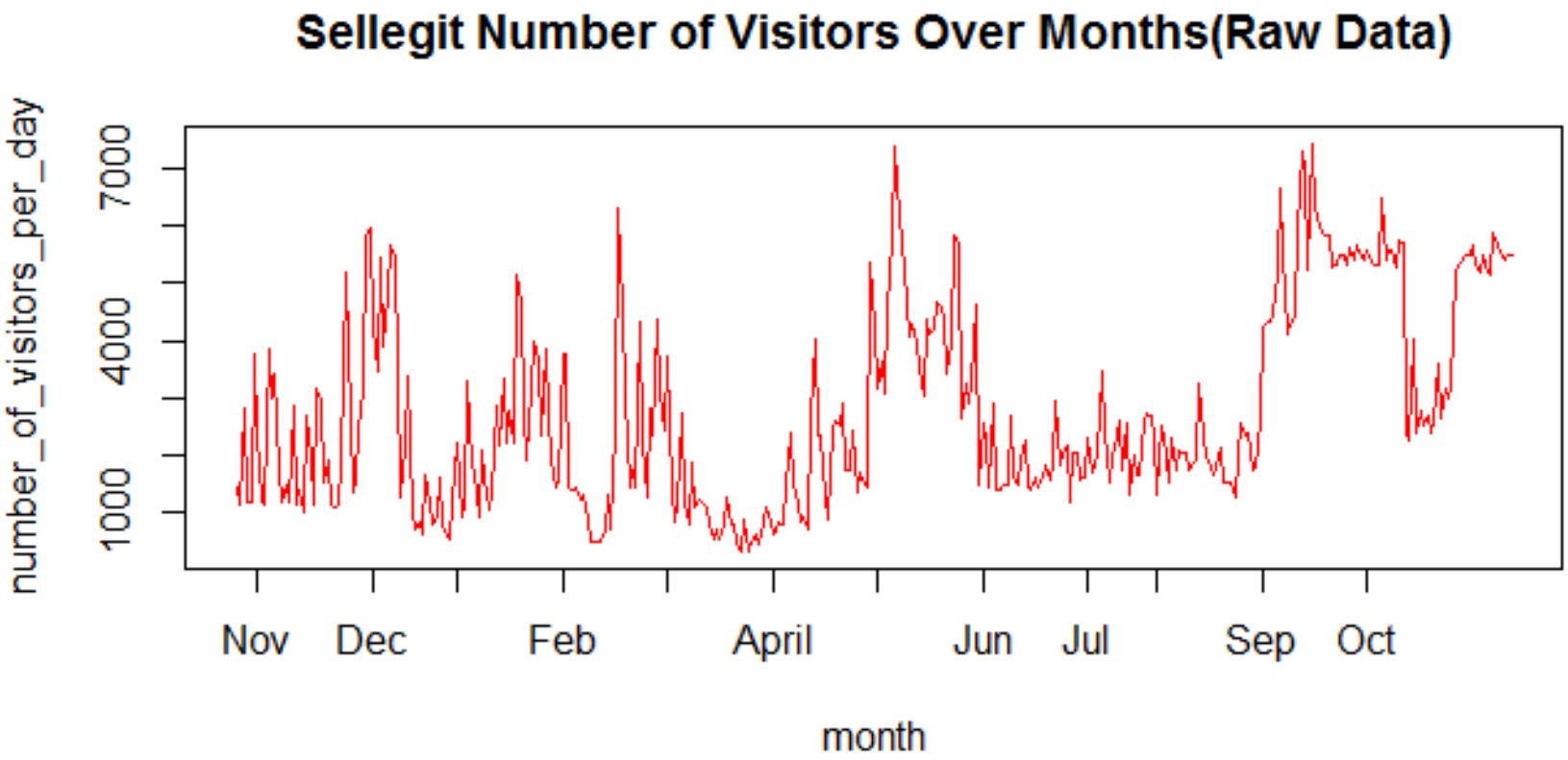
	sar1	constant
	0.6312	3084.9986
s.e.	0.0664	90.6249

Time series model according to our analysis:

$$\hat{Y}_t = 0.6312 + 3085 \cdot Y_{t-a}$$

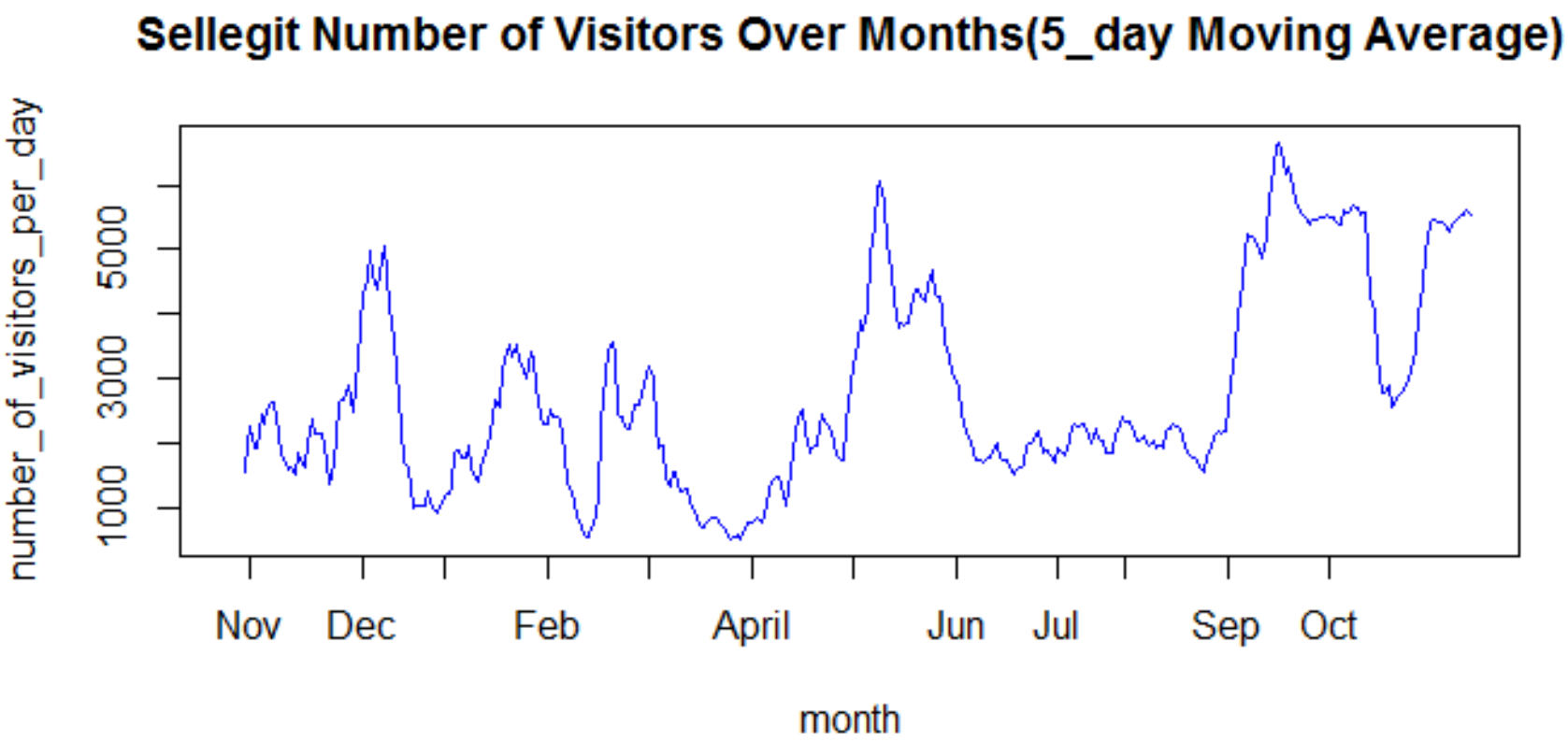
Query 3: Marketing Effort Optimization

R Output:



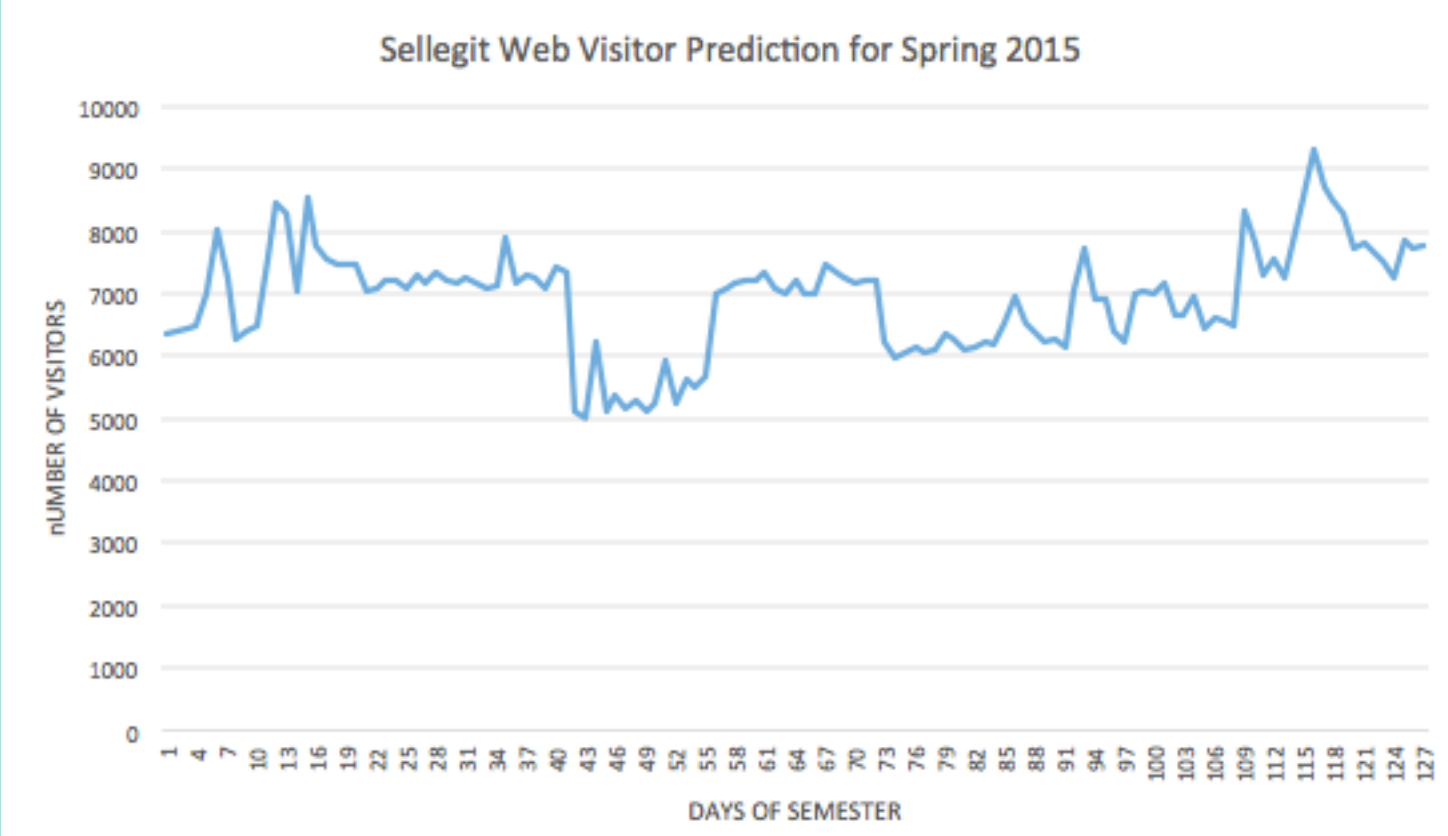
Query 3: Marketing Effort Optimization

R Output:



Query 3: Marketing Effort Optimization

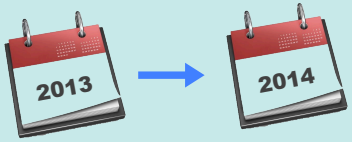
Prediction Output:



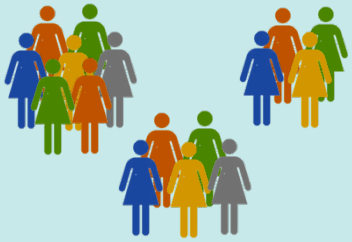
Query 4: User Retention Assessment

How likely will any given user continue to use Sellegit's service after a certain time period?

Calculate retention rate from 2013 to 2014



Analyze how many users bought 1, 2, 3... items last year



Determine survival probability for 2015

Query 4: User Retention Assessment

Business Justification:

- ✓ Forecast future user activity to help **plan company strategy** for expansion and publicity
- ✓ **Reduce client loss** by targeting users who are at the greatest risk of leaving
- ✓ **Identify key events** caused the most increases and decreases in user activity

Query 4: User Retention Assessment

Implementation Process:

SQL

Calculate retention rate from 2013 to 2014

MS Access

Result from SQL query is the input for the Kaplan-Meier Estimators model

Python

Calculate survival probability based on model

Query 4: User Retention Assessment

SQL Code:

```
SELECT Count([Order].UID) AS Expr1
FROM [Order]
WHERE ((([Order].Year)=13));

SELECT Count([Order].UID) AS Expr1
FROM [Order]
WHERE ((([Order].Year)=13));

SELECT Start.Expr1 AS Start, Stay.Expr1 AS Stay,
   ([Stay]/[Start])*100 AS RetentionPercentage
FROM Start, Stay;
```

```
SELECT Order.UID, Count(Order.UID) AS
    CountOfUID
FROM [Order]
WHERE (((Order.Year)=14))
GROUP BY Order.UID;

SELECT HowManyOrderPerUID.CountOfUID AS
   NumItemsSoldPerPerson, Count(*) AS
   NumPeople
FROM HowManyOrderPerUID
GROUP BY HowManyOrderPerUID.CountOfUID;
```

Sample Output:

Start	Stay	RetentionPercentage
53	41	77.3584905660377

NumItemsSoldPerPerson	NumPeople
1	58
2	6
3	3
4	3
5	6
6	1

Query 4: User Retention Assessment

Python Code:

In [1]:

```
import numpy as np
import pandas as pd
import xlrd
data=pd.ExcelFile('Desktop/Shared/sf_Documents/purchase.xlsx')
df=data.parse('Sheet1')
def compute(dataFrame):
    size=list(dataFrame['purchase size'])
    loss=list(dataFrame['number of people'])
    initial=sum(loss)
    SiLoRiSe=zip(size,loss)
    SiLoRiSe=sorted(SiLoRiSe)
    SiLoRiSe[0]=SiLoRiSe[0]+(initial,(initial-SiLoRiSe[0][1])/float(initial),)
    i=1
    while i<len(SiLoRiSe):
      risk_set=SiLoRiSe[i-1][2]-SiLoRiSe[i-1][1]
      loss_set=SiLoRiSe[i][1]
      serivial=SiLoRiSe[i-1][3]*((risk_set-loss_set)/float(risk_set))
      SiLoRiSe[i]=SiLoRiSe[i]+(risk_set,serivial,)
      i+=1
    servival_dict={}
    for element in SiLoRiSe:
      servival_dict['S('+str(element[0])+')']=element[3]
    return servival_dict
out=compute(df)
pd.DataFrame(out,index=[1])
```

Query 4: User Retention Assessment

Python Output:

Out[1]:

	S(1)	S(2)	S(3)	S(4)	S(5)	S(6)
1	0.246753	0.168831	0.12987	0.090909	0.012987	0

Probability of a user purchasing 1 item in 2015

Probability of a user purchasing 2 items in 2015

Probability of a user purchasing 3 items in 2015

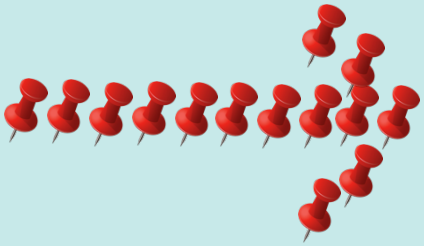
Probability of a user purchasing 4 items in 2015

Probability of a user purchasing 5 or 6 items in 2015

Query 5: Optimal Warehouse Location

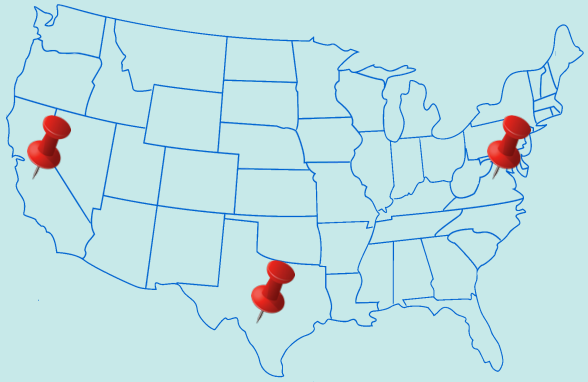
If Sellegit were to build warehouses to store products, where should they build them?

Pinpoint Sellegit user hotspots



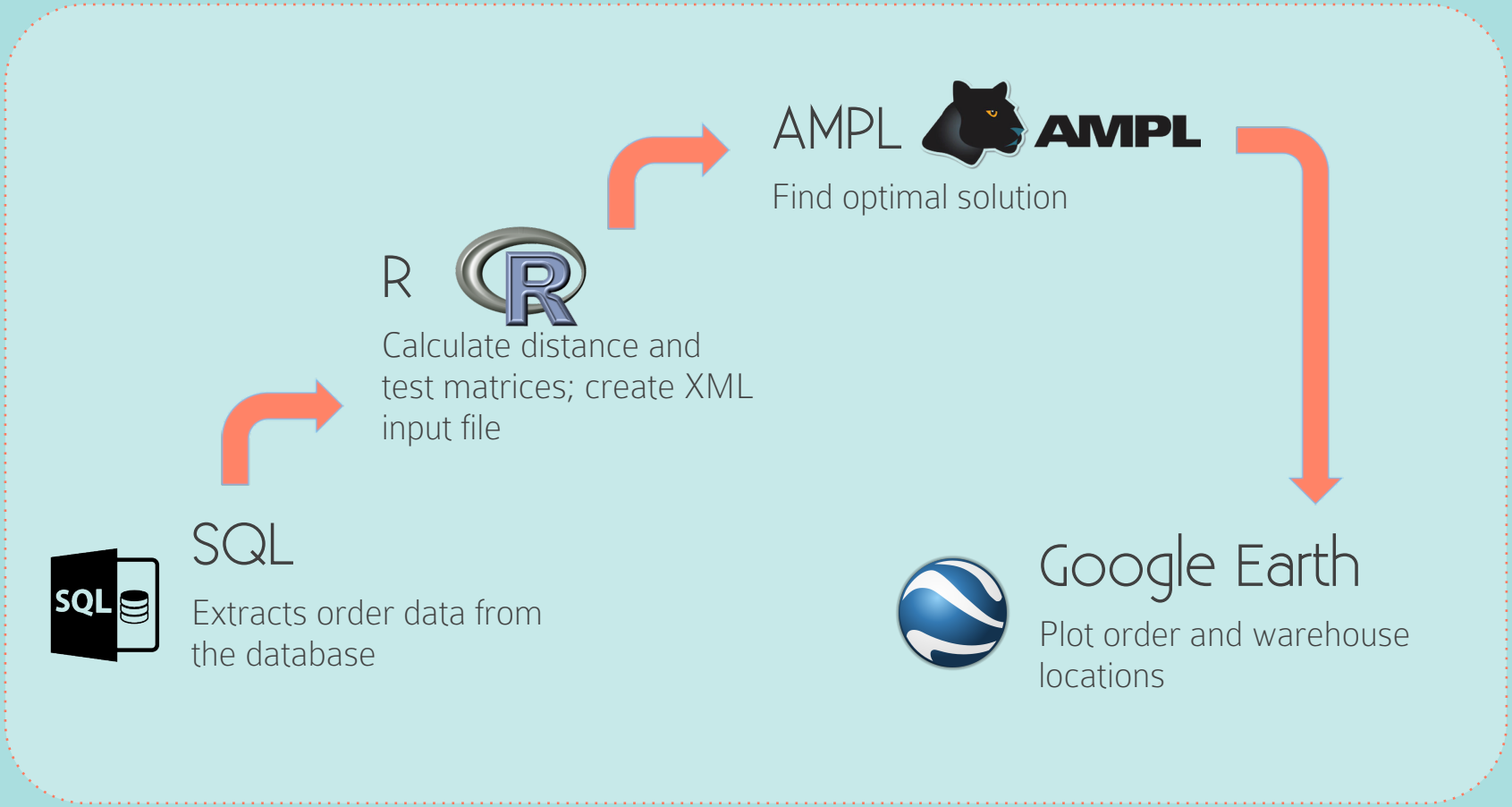
Minimize distance between hotspots

Plot these optimal locations



Query 5: Optimal Warehouse Location

Implementation Process:



Query 5: Optimal Warehouse Location

Business Justification:

- ✓ Optimize company operations by finding the **best central locations** to build warehouses
- ✓ Gain information for **scaling up the company** in future expansion efforts
- ✓ Provide **valuable geographical information** about users that can further help in marketing

Query 5: Optimal Warehouse Location

SQL Code:

```
SELECT city, lati, long
FROM order;
```

Sample Output:

city	lati	long
Berkeley, CA, USA	37.89182	-122.27581
Berkeley, CA, USA	37.86938	-122.27298
Berkeley, CA, USA	37.87005	-122.27848
Berkeley, CA, USA	37.89902	-122.25918
Berkeley, CA, USA	37.88541	-122.25373
Berkeley, CA, USA	37.89512	-122.26083
Berkeley, CA, USA	37.89595	-122.27167
Berkeley, CA, USA	37.8665	-122.27807
Berkeley, CA, USA	37.89064	-122.267
Berkeley, CA, USA	37.90056	-122.26156
Berkeley, CA, USA	37.88279	-122.2767
Berkeley, CA, USA	37.86845	-122.25764
Berkeley, CA, USA	37.88709	-122.26303
Berkeley, CA, USA	37.87925	-122.25835
Berkeley, CA, USA	37.8809	-122.2634

Query 5: Optimal Warehouse Location

R Code:

Load in data to current environment

```
order.data <- read.csv("C:/Users/jie/Desktop/order data.csv")
x = order.data$long
y = order.data$lati
unique(order.data$city)
```

Retrieve the relevant subset of data

```
point = matrix(NA, 183, 2)
point[,1] = x
point[,2] = y
```

Deleting missing data

```
data <- na.omit(point)
```

Set up initial warehouse locations

```
# set up initial points and plot warehouse location
k = matrix(c(-122.24, 37.78, -122.27, 37.83, -122.28, 37.88, -122.42,
37.80, -122.46, 37.78, -122.34, 37.92, -122.2, 37.8, -122.03, 37.53,
-121.96, 37.52), 9,2, T)
```

Query 5: Optimal Warehouse Location

R Code:

Build the distance function

```
earthdist = function(long1, lati1, long2, lati2)
{
  dlong = long2 - long1
  dlati = lati2 - lati1
  a = (sin(pi*dlati/360))^2 + cos(pi*lati1/180) * cos(pi*lati2/180) * (sin(pi*dlong/360)
^2)
  c = 2 * atan2( sqrt(a), sqrt(1-a))
  earthdist = 3961 * c
  return(earthdist)
}
```

Build the Test and Distance matrices for AMPL input

```
test = matrix(0, 9, 183) #binary 0/1
rownames(test) = c("l1","l2","l3", "l4", "l5", "l6", "l7", "l8", "l9")
dist = matrix(NA, 9, 183) #distance
rownames(dist) = c("l1","l2","l3", "l4", "l5", "l6", "l7", "l8", "l9")

for(i in 1:183)
{
  dis = earthdist(k[1:9,1], k[1:9,2], data[i,1], data[i,2])
  dist[,i] =  t(matrix(dis))
  test[which(dis == min(dis)),i] = 1
}
```

Query 5: Optimal Warehouse Location

R Output:

	V1	V2	V3	V4	V5	V6	V7	V8
I1	0	0	0	0	0	0	0	0
I2	0	0	0	0	0	0	0	0
I3	1	1	1	1	1	1	1	1
I4	0	0	0	0	0	0	0	0
I5	0	0	0	0	0	0	0	0
I6	0	0	0	0	0	0	0	0
I7	0	0	0	0	0	0	0	0
I8	0	0	0	0	0	0	0	0
I9	0	0	0	0	0	0	0	0

Test Matrix

	V1	V2	V3	V4	V5	V6	V7	V8
I1	7.97381652	6.43616359	6.57044218	8.29451442	7.32571569	8.03937906	8.20028185	6.33102604
I2	4.28551924	2.72729333	2.80718657	4.80792808	3.93222424	4.52964289	4.56019881	2.5615032
I3	0.84852341	0.82811989	0.69285125	1.73760707	1.4813907	1.47870838	1.19265445	0.93921269
I4	10.1121167	9.35105837	9.11906046	11.1324291	10.8287511	10.8969258	10.46741	9.01053962
I5	12.6843687	11.9364315	11.7050319	13.7080836	13.4145377	13.475399	13.0376594	11.5957506
I6	4.00685568	5.06111634	4.81545619	4.64074393	5.27867879	4.64832417	4.08112198	5.00949064
I7	7.57770843	6.2356475	6.46629533	7.56950367	6.59308423	7.36677785	7.70113517	6.26942809
I8	28.3972867	26.9651643	27.1547697	28.4238026	27.448794	28.2224593	28.5433825	26.9314415
I9	30.9693233	29.6060439	29.8185758	30.8919912	29.9378461	30.7121172	31.081689	29.6068069

Distance Matrix

Query 5: Optimal Warehouse Location

Optimizing Process in AMPL:

Query 5: Optimal Warehouse Location

Model File Code:

$$\text{Min} \sum_{i=1}^{9} \sum_{j=1}^{183} x_i * dist[i,j] * test[i,j]$$

$$\text{S.T} \quad \sum_{i=0}^{9} x_i = 3$$

```
param location > 0 integer; # number of location
param order > 0 integer; #number of order
param dist{1..location, 1..order} >= 0;
param test{1..location, 1..order} >= 0;

var x{i in 1..location} binary; # = 1 if warehouse i is ok, 0 otherwise

minimize distance: sum{i in 1..location, j in 1..order} x[i] * dist[i,j] * test[i,j];
subject to number: sum{i in 1..location} x[i] = 3;
```

Query 5: Optimal Warehouse Location

Data File Code Snippet:

```
param location := 9;

param order := 183;

param dist:     1       2       3       4       5       6       7       8       9       10      11      12      13      14      15
        16      17      18      19      20      21      22      23      24      25      26      27      28      29      30      31
        32      33      34      35      36      37      38      39      40      41      42      43      44      45      46      47
        48      49      50      51      52      53      54      55      56      57      58      59      60      61      62      63
        64      65      66      67      68      69      70      71      72      73      74      75      76      77      78      79
        80      81      82      83      84      85      86      87      88      89      90      91      92      93      94      95
        96      97      98      99      100     101     102     103     104     105     106     107     108     109     110     111
        112     113     114     115     116     117     118     119     120     121     122     123     124     125     126     127
        128     129     130     131     132     133     134     135     136     137     138     139     140     141     142     143
        144     145     146     147     148     149     150     151     152     153     154     155     156     157     158     159
        160     161     162     163     164     165     166     167     168     169     170     171     172     173     174     175
        176     177     178     179     180     181     182     183 :=
1       7.973816522     6.43616359      6.570442178     8.294514417     7.325715692     8.039379065     8.200281846     6.331026039
        7.789585008     8.417321119     7.383269999     6.190177548     7.50942686      6.93417318      7.091522632     7.292854426
        8.073941315     6.867980988     7.861914196     6.70246698      7.653957789     7.930153468     7.392922825     8.239386153
        8.126768049     8.152191506     7.825864483     8.030150953     6.791710001     7.901941923     6.422136295     6.757911364
        6.359814889     6.351641088     6.504401479     6.860309548     8.439578342     6.061566933     8.301450277     7.062690425
        7.496643011     7.410363275     7.571263308     6.380548467     7.049216411     6.604354567     6.481616532     6.899930049
        8.564600669     8.52559494      11.54215747     10.14493599     2.382578525     9.262189488     10.14493599     10.14493599
        9.446551974     11.54215747     11.54215747     11.54215747     13.41140329     11.54215747     9.809121978     10.14493599
        10.14493599     2.382578525     2.382578525     8.137862374     2.382578525     11.54215747     24.26821772     8.137862374
        8.407130357     9.809121978     9.446551974     2.167229709     10.14493599     12.00270226     21.11537595     14.62592989
        11.04531051     3.69105294      10.14493599     9.446551974     8.137862374     9.446551974     11.87758002     2.167229709
        2.187317146     18.19105787     8.866741391     12.27323943     1.026889825     1.026889825     9.446551974     21.11537595
        2.382578525     8.137862374     11.22192294     9.809121978     9.446551974     2.382578525     2.382578525     2.382578525
        13.55904451     2.167229709     2.382578525     2.382578525     2.382578525     2.382578525     2.382578525     2.382578525
        8.137862374     18.19105787     2.382578525     10.14493599     14.79050624     9.446551974     2.382578525     2.382578525
        2.187317146     8.137862374     2.167229709     13.41140329     8.137862374     2.187317146     9.446551974     11.89345179
        11.61376477     11.22192294     11.54215747     11.54215747     9.446551974     2.187317146     8.137862374     14.62592989
```

Query 5: Optimal Warehouse Location

Data File Code Snippet:

```
param test:     1       2       3       4       5       6       7       8       9       10      11      12      13      14      15
        16      17      18      19      20      21      22      23      24      25      26      27      28      29      30      31
        32      33      34      35      36      37      38      39      40      41      42      43      44      45      46      47
        48      49      50      51      52      53      54      55      56      57      58      59      60      61      62      63
        64      65      66      67      68      69      70      71      72      73      74      75      76      77      78      79
        80      81      82      83      84      85      86      87      88      89      90      91      92      93      94      95
        96      97      98      99      100     101     102     103     104     105     106     107     108     109     110     111
        112     113     114     115     116     117     118     119     120     121     122     123     124     125     126     127
        128     129     130     131     132     133     134     135     136     137     138     139     140     141     142     143
        144     145     146     147     148     149     150     151     152     153     154     155     156     157     158     159
        160     161     162     163     164     165     166     167     168     169     170     171     172     173     174     175
        176     177     178     179     180     181     182     183 :=
1       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       1       0       0       0       0
        0       0       0       0       0       0       0       1       0       0       0       0       1       1       0       0
        0       0       0       0       0       0       0       0       0       1       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       1       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       1       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       1       0       0       0       0
        0       0       0       0       0       0       0
2       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       1       0       0       0       0       0       0       0       0       0       0       0
        0       1       1       0       1       0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       1       0       0       0       0       0       0       0
        1       0       0       0       0       1       1       1       0       0       1       1       1       1       1       1
        0       0       1       0       0       0       1       1       1       0       0       0       0       1       0       0
```

Query 5: Optimal Warehouse Location

AMPL Solution Output:

```
ampl: reset;
ampl: model model.txt;
ampl: data data.txt;
ampl: solve;
MINOS 5.51: ignoring integrality of 9 variables
MINOS 5.51: optimal solution found.
5 iterations, objective 40.44984533
ampl: display x;
x [*] :=
1  1
2  0
3  0
4  0
5  1
6  0
7  1
8  0
9  0
;
```

Query 5: Optimal Warehouse Location

R Code for XML Input:

Build the XML node tree skeleton

```
ldoc <- newXMLDoc() # set up XML document
root <- newXMLNode("kml", doc = doc, namespaceDefinitions =
                   "http://www.opengis.net/kml/2.2")
Document = newXMLNode("Document", parent = root)
name = newXMLNode("name", "Sellegit", parent = Document)
description = newXMLNode("description",
                         "Optimized Locations", parent = Document)
```

Populate each coordinate into the above XML tree

```
for (i in 1:length(latitude)) {
  placemark = newXMLNode("Placemark", parent = Document)
  point = newXMLNode("Point", parent = placemark)
  coordinates = newXMLNode("coordinates",
                           paste(longitude[i], ",", latitude[i], sep = ""),
                           parent = point)
  #timestamp = newXMLNode("TimeStamp", parent = placemark)
  #when = newXMLNode("when", time[i],
                    #parent = timestamp)
}
```

Query 5: Optimal Warehouse Location

XML File Snippet:

```
<?xml version="1.0"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Sellegit</name>
    <description>Optimized Locations</description>
    <Style id= "pushpin">
      <IconStyle id= "mystyle">
        <Icon>
            <href>http://maps.google.com/mapfiles/kml/paddle/W.png</href>
         <scale>50</scale>
        </Icon>
        </IconStyle>
    </Style>
    <Style id= "optimal">
      <IconStyle id= "mystyle">
        <Icon>
            <href>http://maps.google.com/mapfiles/kml/pal2/icon10.png</href>
         <scale>8.0</scale>
        </Icon>
        </IconStyle>
    </Style>
    <Placemark>
      <Point>
        <coordinates>-122.27581,37.89182</coordinates>
      </Point>
    </Placemark>
    <Placemark>
      <Point>
        <coordinates>-122.27298,37.86938</coordinates>
      </Point>
    </Placemark>
```

Query 5: Optimal Warehouse Location

XML File Coordinates → Plot in Google Earth

.....

Query 5: Optimal Warehouse Location

Query 5: Optimal Warehouse Location

NORMALIZATION

Decomposing to 1NF

Original Relation:

ItemType(TID, IID, typeName, makeYear, material, brand, madeIn, CoID, faceValue, detailedDescription)

To normalize to 1NF:

A item may be made from many materials, so material is a multi-valued attribute. Let's remove it!

- ItemType(TID, IID, typeName, makeYear, brand, madeIn, CoID, faceValue, detailedDescription)
- Item_Material(IID, material)

Decomposing to 2NF

1NF from previous slide:

- ItemType(TID, IID, typeName, makeYear, brand, madeIn, CoID, faceValue, detailedDescription)
- Item_Material(IID, material)

To normalize to 2NF:

We can know makeYear, brand, madeIn, CoID, faceValue, and detailedDescription from IID.

- ItemType(TID, IID, typeName)
- Item_Material(IID, material)
- ItemInfo(IID, makeYear, brand, madeIn, CoID, faceValue, detailedDescription)

Decomposing to 3NF & BCNF

2NF from previous slide:

- ItemType(TID, IID, typeName)
- Item_Material(IID, material)
- ItemInfo(IID, makeYear, brand, madeIn, CoID, faceValue, detailedDescription)

To normalize to 3NF:

No transitive dependencies!

To normalize to BCNF:

TID → TypeName

- ItemType(IID, TID)
- Item_Material(IID, material)
- ItemInfo(IID, makeYear, brand, madeIn, CoID, faceValue, detailedDescription)

Decomposing to 3NF & BCNF

Original Relation:

Course(CoID, cName, cNum, semester, year, univName)

- Already in 1NF & 2NF

To normalize to 3NF:

cNum, semester, year, univName → cName

- Course(CoID, cNum, semester, year, univName)

To normalize to BCNF:

cNum, semester, year, univName → CoID

- Course(cNum, semester, year, univName)

THANK YOU! ☺

Jeff Zhang

CEO of Sellegit

Peter Chen

Co-Founder of Sellegit

Ken Goldberg

Our FAVORITE Professor!!

Animesh Garg

Our AWESOME GSI!!

Have a wonderful Winter Break
&&
Good Luck with Finals!