Continuous Learning Program

Programming Skills

Do The Math

Chicago, IL
Bangalore, India
www.mu-sigma.com

February, 2011

Agenda

- Structured Programming – what is it?
  - Designing your process
  - Representation of your problem
  - Modularizing your code
- Case Studies & Brainstorming session

Do we want to be code monkeys??

Code Monkey get up get coffee
Code Monkey go to job
Code Monkey have boring meeting
With boring manager Rob
Rob say Code Monkey very diligent
But his output stink
His code not "functional" or "elegant"
What do Code Monkey think?
Code Monkey think maybe manager want to write

god damned login page himself
Code Monkey not say it out loud
Code Monkey not crazy, just proud

Code Monkey like Fritos
Code Monkey like Tab and Mountain Dew
Code Monkey very simple man
With big warm fuzzy secret heart:
Code Monkey like you

Code Monkey hang around at front desk
Tell you sweater look nice
Code Monkey offer buy you soda
Bring you cup, bring you ice
You say no thank you for the soda cause
Soda make you fat
Anyway you busy with the telephone
No time for chat

Code Monkey have long walk back to cubicle he sit down pretend to work
Code Monkey not thinking so straight
Code Monkey not feeling so great
Code Monkey like Fritos
Code Monkey like Tab and Mountain Dew
Code Monkey very simple man
With big warm fuzzy secret heart:
Code Monkey like you
Code Monkey like you a lot

Code Monkey have every reason
To get out this place
Code Monkey just keep on working
See your soft pretty face
Much rather wake up, eat a coffee cake
Take bath, take nap
This job "fulfilling in creative way"
Such a load of crap
Code Monkey think someday he have everything even pretty girl like you
Code Monkey just waiting for now
Code Monkey say someday, somehow

Code Monkey like Fritos
Code Monkey like Tab and Mountain Dew
Code Monkey very simple man
With big warm fuzzy secret heart:
Code Monkey like you

Why is Structured Programming important?

- **Structured Programming makes you more efficient by:**
  - Making it easier to read and debug
  - Over time, enables re-use within your work and across with other teams
- **You will not be the only developer forever …**
  - You will get promoted or move on and somebody will inherit your programs
- **… even if you were, you still can't do it all**
  - Remember, you are part of a bigger eco-system and that could include other developers as well
- **Structured Programming, like structured thinking is 'good' !!**

Design	Represent
Think before you do	**Capture your thoughts**
Think modular, divide and conquer	**Pseudo-code before you code**
Keep it Simple	**Make it readable**

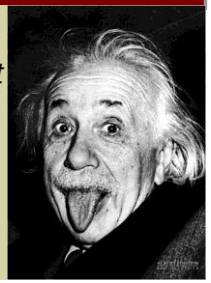
Design should be the single most important phase of your entire programming process

Why is it important?

- Frontload the thinking
- Have conflict sooner
- Avoid rework

"If I had one hour to save the world, I would spend the first 55 minutes defining the problem."

- *Albert Einstein*
- *Winner, Nobel Prize in Physics, 1921*



Think before you do

Why precedes how: Understand the why and never lose sight of that
Draw the big picture: White-board the overall solution and use this to drive your plan
Overall System view: Draw an end-to-end flow of what your program needs to do

Think Modular

Separation of concerns: Each module has minimal overlap with other modules.
Encapsulated: Communicates through a well defined interface
Substitutable: can be replaced by another module with minimal impact on the system.
Reusable: Can be used in other programs

Keep it Simple …

Representation is key to making your programs transparent to others

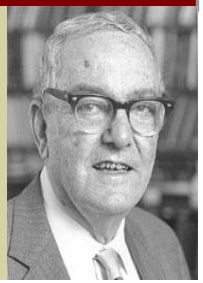
Why is it important?

- Visualize the problem
- Communicate better
- Collaborate better

"Solving a problem simply means representing it so as to make the solution transparent."

- Herbert A. Simon

Winner, Nobel Prize for Economics, 1978



Capture your thoughts

Get the flow right: Capture the logical flow that each program needs to follow
Get the objectives right: Capture what you want the program to do
Get the outputs right: Capture what you want to get out of the program

Pseudo-code before you code

Create an ordered sequence: Number the steps
Keep it unambiguous and well-defined: clear, do-able, and can be done without difficulty
Make sure it performs some task
Make sure it halts in finite time: All algorithms must terminate!

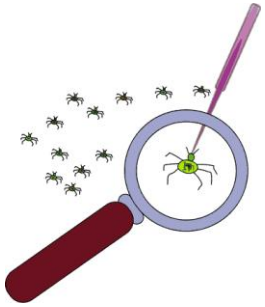
Make it readable …

Modularizing your code will allow you to break the big task into smaller tasks

"Debugging is twice hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

Brian Kernighan

Modular Code



DEBUG

UPGRADE

REUSE

Instructions

- Structure of your output:
  - Define the Objective(s)
  - List down your assumptions
  - Develop a workflow
  - Document the programming flow in the form of pseudo-code.
- Do not use Powerpoint or Excel – the work has to be done in Word.
- **Special focus needs to be on:**
  - **Design thinking**
  - **Modularity**

REMEMBER: THE PRIMARY FOCUS OF THIS ASSIGMENT IS TO EXPLORE DESIGN THINKING AND MODULARITY AND NOT ON ALGORITHMS !!!

Case Studies

- Customer Purchase Pattern
- First Purchase Product Mix
- Product Sales Analysis
- Product Pricing Analysis

Case Study 1 – Customer Purchase Pattern

Study of customer purchase pattern across different months

- **Problem :** To compute cumulative sales & time since previous purchase (in months) for every customer studying for every month starting from their respective registration month to May'11

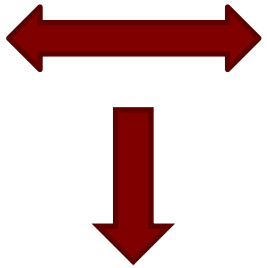
- **Background** : Retail client is an online subsidiary. Customers are assigned a customer id when they login or register on the website and that date is called as the registration date of that customer. Of all the customers who register, some of them purchase and their data is recorded in the transaction data.

- **Datasets required for the analysis** :
  - **Registration data** (PK – Customer id) : Registration date/month of each customer
  - **Orders data** (PK – Customer id, Order month) : details of orders made rolled up at a customer, month level.

Snapshot of Input & Output data

Registration data layout	
Customer_id	**Reg_month**
a	11/1/2010
b	6/1/2010
c	1/1/2010

Note : c is a registered customer, but did not make a single purchase during his tenure



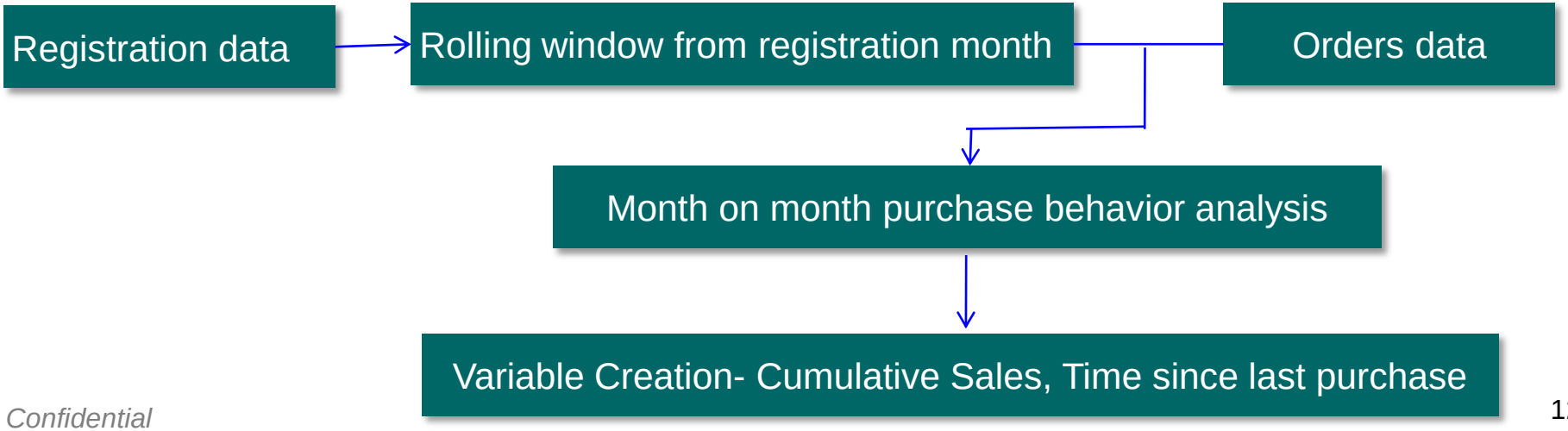
Order data Layout			
Customer_id	**Order_month**	**Sales ($)**	**Orders**
a	12/1/2010	20	2
a	2/1/2011	30	5
a	4/1/2011	50	6
b	10/1/2010	10	1
b	12/1/2010	12	4
b	5/1/2011	45	5
b	6/1/2011	23	2

Output table layout						
customer_id	**reg_month**	**observation_month**	**order_month**	**Sales**	**Time_since_prev_purchs****	**Cum_sales***
a	11/1/2010	12/1/2010	12/1/2010	20	0	0
a	11/1/2010	1/1/2011	.	.	1	20
a	11/1/2010	2/1/2011	2/1/2011	30	2	20
a	11/1/2010	3/1/2011	.	.	1	50
a	11/1/2010	4/1/2011	4/1/2011	50	2	50
a	11/1/2010	5/1/2011	.	.	1	100
a	11/1/2010	6/1/2011	.	.	2	100
b	6/1/2010	10/1/2010	10/1/2010	10	0	0
b	6/2/2010	11/1/2010	.	.	1	10
b	6/3/2010	12/1/2010	12/1/2010	12	2	10
b	6/4/2010	1/1/2011	.	.	1	22
b	6/5/2010	2/1/2011	.	.	2	22
b	6/6/2010	3/1/2011	.	.	3	22
b	6/7/2010	4/1/2011	.	.	4	22
b	6/8/2010	5/1/2011	5/1/2011	45	5	22
b	6/9/2010	6/1/2011	6/1/2011	23	1	67

Cum_sales* : cumulative sales excluding the current month (observation month) value
Time_since_prev_purchs** : Time since previous purchase in months

Process Flow

- A random customer sample was selected for the study
- A dataset was created studying all the customers for all the months starting from their registration month to May'11 (For eg: Customer registered in Jan 2011, will have 6 rows in the dataset for every month till May 2011)
- The order data was then merged with the above rolling dataset to get month by month purchase activity of the customer
- This was then utilized to find 'Time since last purchase' and cumulative sales for every



The techniques used for the study

- Functions like intck, intnx were used to create flags and months for the rolling window
- Internal do loop- The do loop was used to create the rolling month on month window for analysis
- By statement- This was used to process the dataset in groups of customer id
- Retain statement – This was used to calculate the value of Time since last purchase and cumulative sales

Case Study 2 – New Customers 1st purchase product mix

Study of first purchase LOB mix for new customers

- **Problem :** To study the first purchase LOB mix pattern for new accounts during the last five quarters.

- **Background** : Technology based client is an American multinational information technology corporation that develops, sells and supports IT related products and services. Client wants to understand the product preference of the new customers

- **Datasets required for the analysis** :
  - **Transaction Data(**PK- Account ID, order date, product) **:** Daily purchase data of all the customers for a specified period

Snapshot of Input & Output data

INPUT

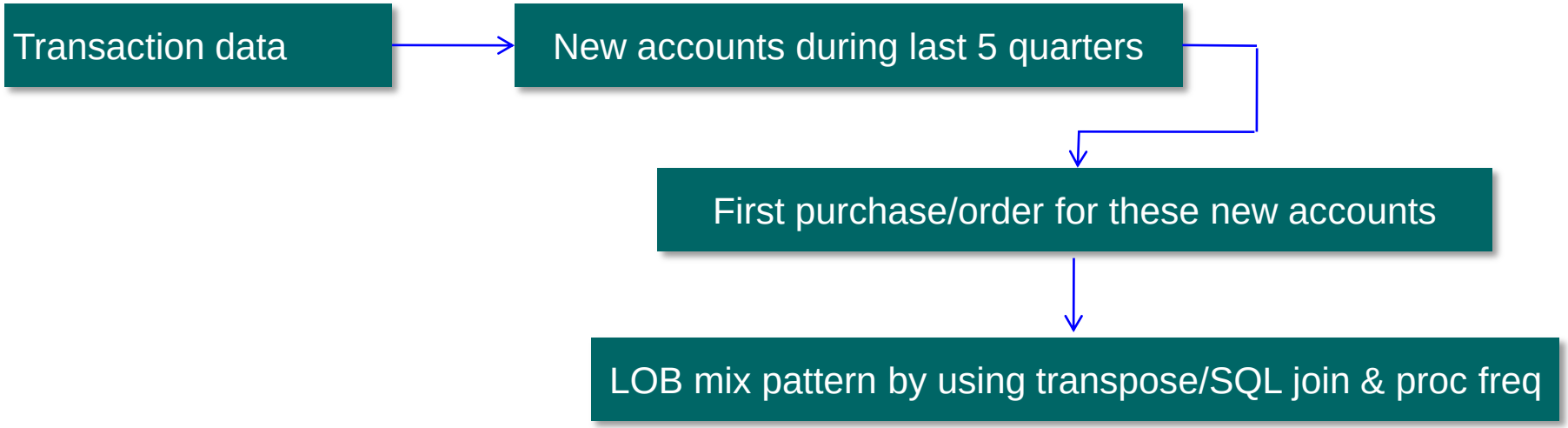
	ACCOUNT_ID	ORDER_DATE	GROUP_DESC	PRODUCT_DESC	TRANSACTION_DATE_QUARTER	TOTAL_AS_SOLD_REVENUE_USD	SALES_MARGIN_USD	SYSTEM_QTY	L
1	1938224897	06/16/2010	Non_Tied Peripherals	Non_Tied Peripherals	2011Q02	428.179	18.064	0	
2	1938224897	06/17/2010	Non_Tied Peripherals	Non_Tied Peripherals	2011Q02	1184.416	40.773	0	
3	1938224897	06/17/2010	Desktops	OptiPlex Desktops	2011Q02	771.807	183.36	1	
4	1938224897	02/10/2011	Desktops	Vostro Desktops	2012Q01	1023	346.755	1	
5	1938224897	02/21/2011	Desktops	OptiPlex Desktops	2012Q01	684.059	165.682	1	
6	1938224897	03/30/2011	Notebooks	Vostro Notebooks	2012Q01	983.275	218.144	1	
7	1938224897	03/31/2011	Desktops	OptiPlex Desktops	2012Q01	3042.845	810.063	3	
8	1938224897	06/09/2011	Desktops	OptiPlex Desktops	2012Q02	1039.268	328.992	1	
9	1938224926	11/06/2009	Desktops	OptiPlex Desktops	2010Q04	1463.146	549.687	1	
10	1938225051	04/19/2005	Desktops	Personal Desktops	2006Q01	631.954	577.998	1	

Output

	Fixed Wk	Latitude	Mobile Wk	NTP	OptiPlex DT	Personal NB	Power Edge	Vostro DT	Vostro NB	XPS NB
Fixed Wk	66%	0%	2%	26%	4%	0%	0%	2%	0%	0%
Latitude	0%	55%	0%	29%	8%	0%	0%	4%	4%	1%
Mobile Wk	10%	0%	70%	20%	0%	0%	0%	0%	0%	0%
NTP	2%	5%	0%	63%	5%	0%	3%	10%	10%	0%
OptiPlex DT	1%	5%	0%	17%	72%	0%	1%	3%	2%	0%
Personal NB	0%	0%	0%	30%	0%	60%	0%	0%	10%	0%
Power Edge	0%	0%	0%	25%	1%	0%	67%	1%	0%	0%
Vostro DT	0%	1%	0%	14%	1%	0%	0%	78%	5%	0%
Vostro NB	0%	2%	0%	20%	1%	0%	0%	7%	70%	0%
XPS NB	0%	10%	0%	30%	0%	0%	0%	0%	0%	60%

Process Flow

- The customer transaction data for the period Q3FY11 to Q3FY12 is considered.
- A dataset was created studying all the new accounts during the 5 quarters starting from Q3FY11 to Q3FY12.
- Then only the first purchase for the new accounts is considered leaving alone the rest of the purchases.
- This dataset was then utilized to find the LOB mix pattern for those set of accounts.



The techniques used for the study

- By statement was used to process the dataset in groups of Account id
- Functions like transpose/SQL joins were used to get the different LOB/product bought in 1st purchase
- PROC FREQ – was used to get the required cross tab along with Row % of accounts, thus getting the final required table in the deck

Case Study 4 – Product Pricing Analysis

Study of pricing of items in each category

- **Problem :** To calculate the percentile of the AUR (Average Unit Retail, AUR = (Sales/ Units Sold) of items in a category and bucket the items based on the AUR. The buckets are <25 Percentile, 26 – 75 Percentile and > 76 Percentile.

- **Background**: Client is the world's biggest retailer and is under an aggressive pricing initiative, it needs to ensure that pricing of its items are lower when compared to competitors. For doing this, it needs to understand the pricing at a category level for all the items.

- **Tables required for the analysis** :
  - **Transaction Data :** Daily sales data from all the stores for a specified period
  - **Item dimension:** Mapping the items in the transaction table to the levels in the merchandise hierarchy

Process Flow

- Take a period for which item pricing has to be studied
- A table is created for all the transactions in the specified period, the item dimension table is joined with to get the department and category of the items sold
- The transactions are then rolled up at a department, category, item level and the AUR is calculated using the aggregated sales and units sold
- The items for each category are sorted based on the AUR and the row numbers are calculated.
- For each category, calculate the maximum value of the rownumber(100^{th} Percentile).
- The Percentile is then calculated for each category

  Item AUR Percentile = (Row Number of item – 1)/ (MAX AUR in category -1)
- After the AUR percentiles have been calculated for all the items, the total number of items for each bucket is then computed using a CASE statement

The techniques used for the study

- Aggregate function SUM is used for the calculation of the AUR, MAX is used to get the MAX row number (100th Percentile) for each category
- Analytical function ROW NUMBER is used to calculate the position of each item in the category and then for the computation of the percentile
- ORDER BY clause is used to sort the rows based on the AUR for each category

Case Study 5- Order Allocation Process

Optimize the order allocation process for a retail client having an online subsidiary

- **Problem :** To determine the store that will cater the order amount, out of all the stores that are eligible for the order allocation by optimizing the process based on total units assigned

- **Background** : Retail client is an online subsidiary. All the orders made are online and will be shipped to home via a store. Certain rules :
  - If there are multiple stores that can fulfill an order, assign the order to the store with the lowest assigned unit quantity
  - An order amount cannot be divided among different stores. One store will fulfill the whole order amount
  - If there are two stores that can fulfill the order and both the stores have catered equal amount of orders so far, then that order can be fulfilled by either of the stores.

- **Datasets required for the analysis** :
  - Order Allocation data (PK – Order number) : Information of the order amount and stores eligible to cater the order

Snapshot of Input & Output data

Order Allocation - Input dataset		
Ordernum	Orderamt	StoresEligible
1	20	S3
2	12	S1
3	19	S1,S2
4	7	S1,S2
5	14	S1,S2,S3
6	19	S2
7	5	S1,S2
8	17	S2
9	13	S3
10	5	S1
11	2	S1,S2,S3
12	20	S1,S3
13	3	S1,S3
14	3	S1,S2
15	16	S1
16	10	S1,S2,S3
17	11	S2
18	4	S3
19	7	S1,S2,S3
20	19	S3

Order Allocation Index - Output Dataset					
ordernum	orderamt	load_s1	load_s2	load_s3	store_idx
1	20	0	0	20	3
2	12	12	0	20	1
3	19	12	19	20	2
4	7	19	19	20	1
5	14	33	19	20	1
6	19	33	38	20	2
7	5	38	38	20	1
8	17	38	55	20	2
9	13	38	55	33	3
10	5	43	55	33	1
11	2	43	55	35	3
12	20	43	55	55	3
13	3	46	55	55	1
14	3	49	55	55	1
15	16	65	55	55	1
16	10	65	65	55	2
17	11	65	76	55	2
18	4	65	76	59	3
19	7	65	76	66	3
20	19	65	76	85	3

*Assume that the initial load is 0 for all stores.