Data Mart Teams

- We already have teams that built stores 1-10
- Now we'll roll those up into regions A=1-4, B=5-7, C=8-10

Key Steps

- Step 1 - Data Profiling: You will want to do a series of profiling steps to make sure your store databases are reasonable
  - For example if Froot Loops are the 10th best selling item in store 1, but they're the 900th ranked item in store 2 then there is something wrong with one of the programs.
  - (By the way, each store should have a name, and the individual team member(s) who generated the data will be listed as the store manager (e.g. the Bryan-Johann Market). You can make up an address and other info as shown in the table several slides ahead.)
  - Also, profile sales, customers per day, total transactions and make sure differences among stores make sense

Key Steps (2)

- Step 2 – Fix individual programs
  - If there are problems with one or more programs, then team members should help other team members fix any bugs
  - Do not just replace all programs with the best working one. We want different programs to have different output formats etc. to mimic how actual chains of stores work. So, fix the existing programs unless they're completely hopeless.
    - The idea here is that chains of stores will have scanners from different manufacturers or different eras and will consequently have different feed formats. ETL teams must deal with data in multiple formats
  - Next re-run profiling exercise.
    - Check for missing values, missing dates, etc.
    - Recheck top selling items as before
    - Deliverable 1 should look something like the next slide

Deliverable One

Combines Sales (December)			Joe's Market		Sue's Market		Jane's Market		Steve's Market	
$ 13,043,675.00			$ 3,069,100.00		$ 3,682,920.00		$ 3,376,010.00		$ 2,915,645.00	
Combined Customer Count (December)			20400		24480		22440		19380	
Best Selling Items Combined		# Sold	# Sold	Rank	# Sold	Rank	# Sold	Rank	# Sold	Rank
SKU 1	Product 1	130050	30600	1	36720	3	33660	2	29070	1
SKU 2	Product 2	127415	29980	2	35976	2	32978	1	28481	3
.										
.										
.										
SKU 25	Product 25	63707.5	14990	25	17988	24	16489	28	14240.5	23

This will be an actual product name like Whole Milk

Note the rankings will differ by store, but not wildly.

A Common ETL Issue

- Missing Data/Conforming Data – The products table has multiple issues
  - Some ItemTypes are missing
  - Some ItemTypes are too broad
  - Some ItemTypes are too fine

- Upper Management has decided that every SKU should be mapped to one of the 114 product subcategories in product_class.txt

- This happens all the time. Companies get reorganized into new business units and the Data Warehouse must get reorganized as well
  - It's a pain, but remember a Data Warehouse is for business users in management. They don't care about inconveniencing the ETL team

Fixing the Products Table

- The Good News: 1319 of the 2076 products have an itemType that maps directly to the subcategory

- The Bad News: 294 items in the product table have null values. These will have to be assigned to one of the new subcategories
  - is there a way to automate this, by string searching or is it a data entry problem? ETL team decides.
  - Are there any products that don't fit into a category? Suggest a new category to management (me) if you find one that doesn't fit.

- More bad news: There are 597 items with item types, but those item types are obsolete (either too broad or too fine).
  - For example there are a bunch of Potato Chips currently called 'Snacks' that want to go to 'Chips' and a bunch of 'Bread' that want to go to 'Sliced Bread'
  - I'll bet there is a way to automate these changes

Deliverable 2a

- Deliverable 2a: A new products table that contains: Manufacturer, Product Name, SKU, Size, Product Class ID, etc as shown
  - This should have no null values (and only Product Class IDs that exist in the Product Class Table or new ones that Management has agreed to add)

Product Dimension	
Product key	int
SKU	ShortText
Product Name	ShortText
Product Class ID	int
Subcategory	ShortText
Category	ShortText
Department	ShortText
Product Family	ShortText
Size	ShortText
#Per Case	int
Brand Name	ShortText
Manufacturer	ShortText
Supplier	ShortText

- Note supplier is always Rowan Warehouse for everything but milk. Milk comes from Rowan Dairy
- Note the product key is a random integer for each SKU. (You might want to ask other teams how they are generating theirs, otherwise when we roll up all the data-marts into an Enterprise Data Warehouse, we will have an issue)

Deliverable 2b

- Deliverable 2b: We haven't talked about meta-data much, but every record needs a source and a reason.
  - So we'll need another table that contains each SKU and source#
  - Source# can just be 1,2,3,4,5, etc. where 1 means it came from original product table, 2 means it was mapped by hand by Jane Doe, 3 means it was done by a string match like Product Name = 'Frito Lay' implies subcategory 'Chip' etc.
  - You will obviously need a source# table that contains definitions similar to above for each source#

Another Common ETL Problem

- It's likely that if you have 3 stores that you may have more than one date format (e.g. 20240101, 1/1/2024, January 1, 2024
- Make sure as part of your ETL process you conform the dates.
- Don't rewrite the individual programs to use the same format (that's cheating). Instead conform the dates in the staging database that combines the store data
- You might ask the other teams what kind of date format they're using so we don't have to do this again when we roll-up the data-marts into an Enterprise Data Warehouse (Maybe we should just go with 20240101)
- Note this is why Bill Inmon (father of data warehousing) doesn't like building Data Warehouses bottom-up. He believes the Data Warehouse should be built first, and then individual data marts should be subsets (But not everyone has 20 million dollars, so most of the time they are built bottom-up like ours)

Deliverable 3

- This one's an easy one

Store Dimension	
Store key	int
Store Manager	ShortText
StoreStreetAddr	ShortText
StoreTown	ShortText
StoreZipCode	ShortText
StorePhone#	ShortText
StoreState	ShortText

- Store key is on an earlier slide
- Manager name is one of you
- Other fields can be made up

Deliverable 4

- Another easy one

Date Dimension	
DateKey	Int
Date	Date/Time
DayNumberInMonth	int
DayNumberInYear	int
WeekNumberInYea	int
MonthNum	int
MonthTxt	ShortText
Quarter	int
Year	int
Fiscal Year	int
isHoliday	Boolean
isWeekend	Boolean
Season	ShortText

- I would make the datekey 1 to 365. (a 2-byte int should allow for 50 years so that ought to be enough)
- Someone will have to look up the holidays and fill in the appropriate fields
- Base seasons on the Solstices and Equinoxes
- Fiscal Year ends in July so anything before July 31 is 2023 and anything after is 2024 (It's common for companies to have fiscal years independent of calendar year because hardly any companies are founded on January 1)

Deliverable 5

- Not Really a deliverable (don't send it to me)

Sales Fact (Transaction Level)	
DateKey	Int
DailyCust#	Int
ProductKey	Int
StoreKey	Int
QuantitySold	Int
TotalDollarSales	float
TotalCostToStore	float
GrossProfit	float

- Just do it for the month of December

- Total dollar sales is sale price*quantity; Total cost is base price*quantity; profit is Total Dollar Sales – Total Dollar Cost

- DateKey,DailyCust#,ProductKey is your composite key (it's unique unless something went wrong) and each component links back to the corresponding dimension tables with no integrity issues

Deliverable 6

- This one is a daily aggregate of the individual transaction table

Sales Fact (Daily Level)	
DateKey	Int
ProductKey	Int
StoreKey	Int
#SoldToday	int
CostOfItemsSold	float
SalesTotal	float
GrossProfit	float

- Note you can't use your Deliverable 5 for this though because that only has a month's data

- A better approach would be to aggregate from the individual stores and then append them together

Deliverable 7

- Not really a deliverable. I don't want the whole table

InventoryFact (Daily Level)	
DateKey	int
ProductKey	int
StoreKey	int
#Available	int
CostToStore (itemLevel)	float
CostToStore (caseLevel)	float
#CasesPurchasedToDate	int

Deliverables 8-11

- 4 Quarterly inventory snapshots (these we want)

InventoryFact (Quarterly Snapshot)	
ProductKey	int
StoreKey	int
Quarter and Year	ShortText
Quarter	int
Year	int
#CasesPurchasedToDate	int
#CasesPurchasedThisQuarter	int
#CasesOnHand	int
TotalCostToStoreThisQuarter	float
TotalSoldByStoreThisQuarter	float
TotalCostToStoreYTD	float
TotalSoldByStoreYTD	float