Structuring Data for Customer Insights

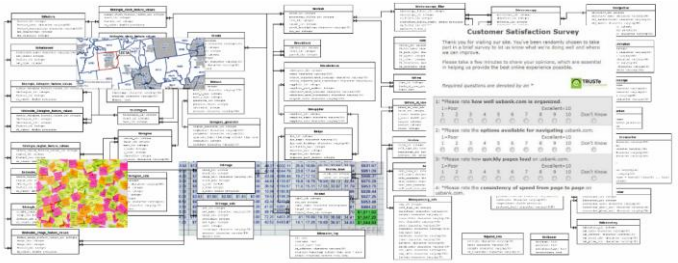
by

Business Analysts and Data Scientists

Jim Porzak
Customer Insight &
Analytics Summit
Austin, TX
August 24, 2016

What We'll Cover

Customers



Data

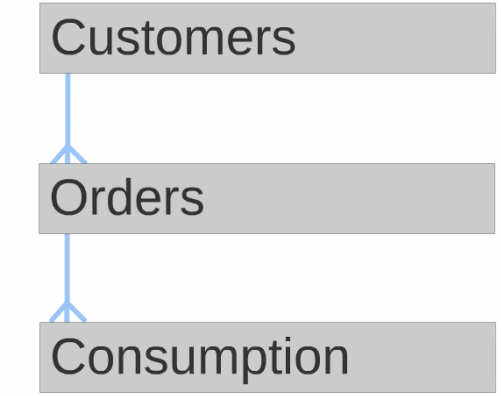
Data Analysts

SQL Query Examples
- Subscription Business -

A. How many subscribed customers do we have?

APPENDIX

1. Real-world subscription business example.
2. Sessionization in SQL and R



Structured Data for Customer Insights

Business Analysts

Who Are They?

Anyone who gives you money; or

has ever given you money; or

should give you money.

Very broad!

Regardless of the nature of your organization!

- Product sales
- Subscription sales
- Charity
- ???

Why do we care?

Advice to Jim: "Take care of your customers and they will take care of you" – Tony Gault, 1983

"We see our customers as invited guests to a party, and we are the hosts. It's our job every day to make every important aspect of the customer experience a little bit better." Tweet this quote.

— Jeff Bezos, Founder & CEO of Amazon

"Your website isn't the center of your universe. Your Facebook page isn't the center of your universe. Your mobile app isn't the center of your universe. The customer is the center of your universe." Tweet this quote.

— Bruce Ernst, VP of Product Management at Monetate

Be Customer Focused!

Take Care of Them!

To do so, we need to understand them.

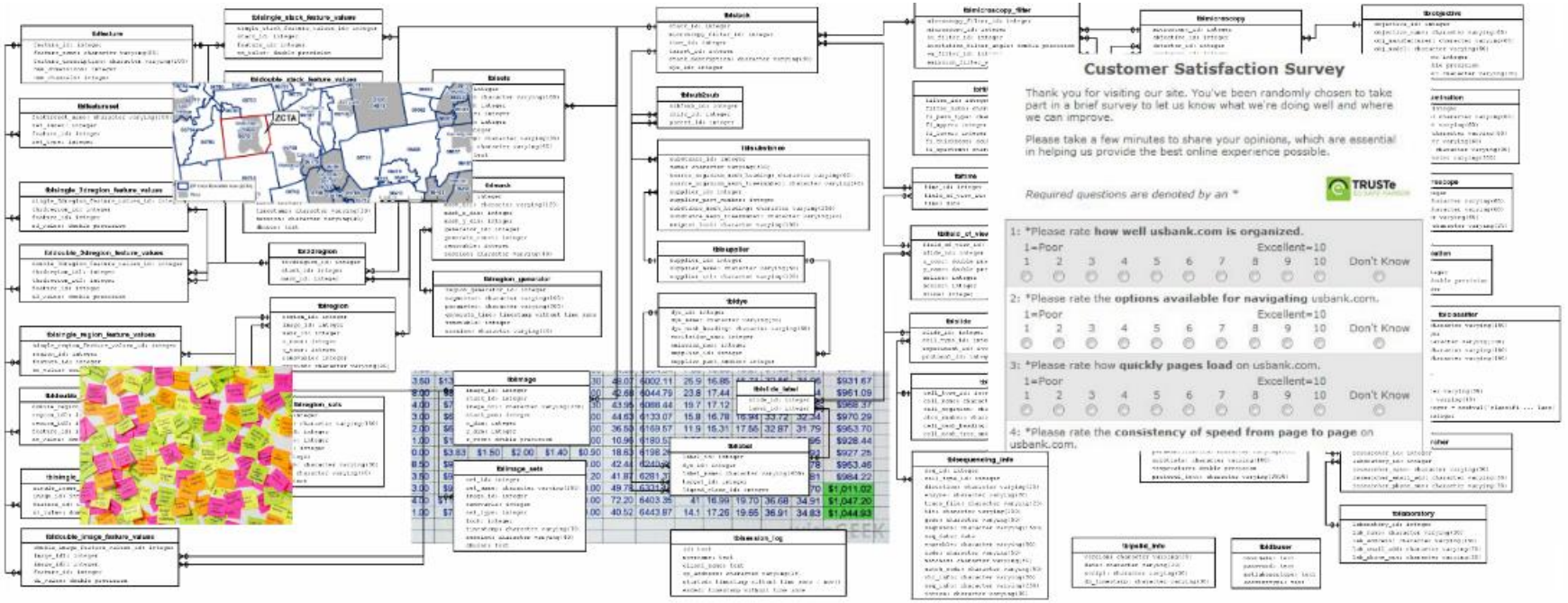
Back to Journalism 101!

Five of these are kind of easy.

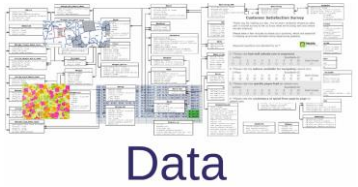
The Big Customer Challenge

Why?

Customers Are Viewed Through…



Data



Data Flavors

Operational Systems

- Order Processing
- Fulfillment
- Web site(s)
- Call Centers (CRMs)
- Email system(s)
- Front line employee “systems”

Direct Responses

- Stated preferences
- Search requests
- Mindful clicks
- Surveys: NPS, satisfaction, …
- Social Comments

Customer ID!

Data Appends

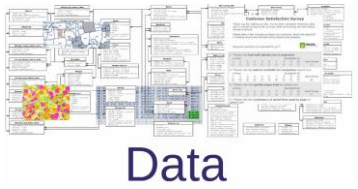
- Neighborhood demographics
  - US Census @ ZIP5 or Census Block Group level
  - PRIZM, etc
- Individual match-backs: Acxiom
- Pooled industry data

Analytic Results

From our data science team:

- Segments
- Scores
- Recommendations

- *Updated in “real time” – relative to customer’s timescale.*

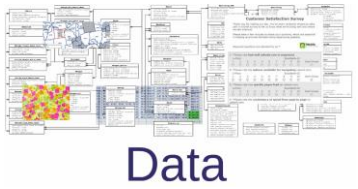


Op Sys are Problem Focused!

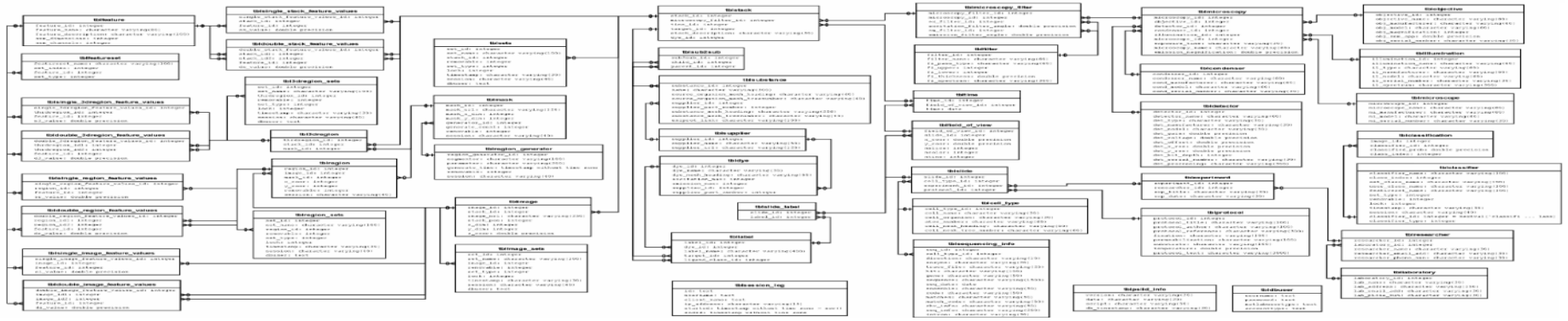
How do you sell bread?

1. You tell me what you want.
2. I give you the bread.
3. I tell you what it costs.
4. You give me the money.

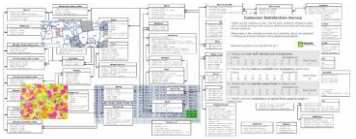
Who?
What?
When?
Where?
Why?



Op Sys are Complex!



- Designed for operational performance.
  - Hopefully 3rd normal form, but …
- Often multiple ways to join tables.
- Columns not used as intended, overloaded, or their use changes over time as needs evolve.
- ***Customer focused columns, if any, seldom QA'd!***



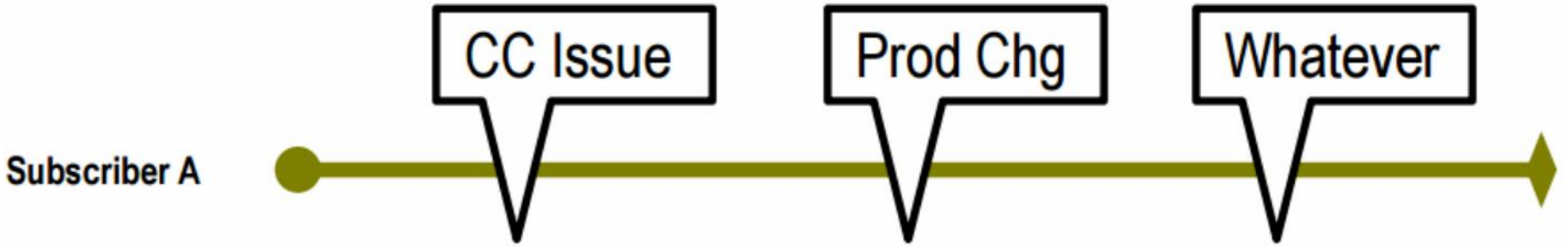
Op Sys are Noisy!

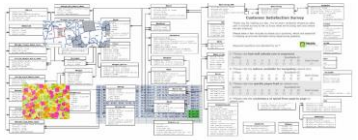
Many op sys events are irrelevant from the customer insights perspective.

Example: Subscription processing needs to deal with the real world and handle payment exceptions which don't reflect customer decisions.

The subscriber decides to start and, possibly, stop:

But the subscription op sys reports:





Data Flavors - Review

Operational Systems

- Order management
- Fulfillment
- Web site(s)
- Call Centers (CRMs)
- Email system(s)
- Front line employee "systems"

Everything else is pretty straightforward – often single files.

Customer ID!

Direct Responses

- Stated preferences
- Search requests
- Mindful clicks
- Surveys: NPS, satisfaction, …
- Social Comments

Data Appends

- Neighborhood demographics
  - US Census @ ZIP5 or Census Block Group level
  - PRIZM, etc
- Individual match-backs: Acxiom
- Pooled industry data

Analytic Results

From our data science team:

- Segments
- Scores
- Recommendations

- *Updated in "real time" – relative to customer's timescale.*

Data Related Roles

Data Engineers

The data geeks responsible for knowing & maintaining the data generated by your organization.

- Typically in IT, data services, or finance
- Power SQL gurus; some data architecture skills
- Big data platform skills.
- Data schlepping & ETL
- Engineering mindset: QA/QC, git, agile, and (hopefully) documentation
- Have overflowing work queue!
  - Mission critical stuff
  - *Ad Hoc* from business

Business Analysts

The front line business folks tasked with understanding your customers and, maybe, communicating with them.

- Embedded in departments
- Results focused
- Time pressure
- Domain experts
- Probably not titled “analyst”
- Excel & PowerPoint
- Data exploration/reporting: Tableau, QlikView, etc
- Maybe lite SQL, R, ?
- Constantly bug data engineers!

Data Scientists

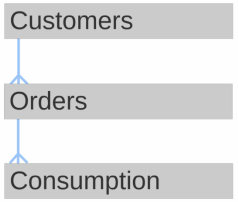
Responsible for deep insights, propensity models, segmentation, and next best action scores.

- Should be close to business users & have strong domain knowledge.
- Strategic & Tactical roles
- Machine Learning skills in R, Python, or?
- Understand statistics & probability.
- May do their own data prep - if they do, ~ 80% of effort
- Predict() results may be operationalized.
- Scarce resource!

A Data Structure for Customer Insights

Basic design principles:

1. Be Analyst Ready.
2. Front end tool agnostic.
3. Model Customer Decisions.
4. Has three Levels of Abstraction:



1. Be Analyst Ready

"Ready" in this context means the user must be really comfortable with this data source. Goals:

- Easy – technically & conceptually
- Self documenting
- Complete
- Accurate
- Timely
- Fast
- Evolving

2. Front-end Tool Agnostic

If this data design is to serve a wide range of data consumers, ranging from junior business analysts to grey haired data scientists, *it must be totally tool agnostic.* Implementation must support:

- Native SQL access
  - JDBC preferred; ODBC if needed.
- Optimized drivers for top tools like Tableau, …
- Decent Excel performance.

Jim's last product platform was on PostgreSQL, last subscription platform on AWS Redshift.

3. Model Customer Decisions

Focus on **CIA**'s

Customer Initiated Actions - and not site or application generated events which just add noise.

Think like a customer when designing your model!

Don't get distracted by all your cool technology, tools, and marketing theories!

4. Three Levels of Abstraction

Each level represents “all there is to know” about the level – an evolving goal, not initial deliverable!

The general terms are:

1. Customers
2. Orders
3. Consumption

Rename appropriately for your business.

Levels of Abstraction – Two Flavors

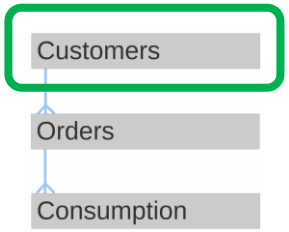
Subscription Business (SB)

1. Subscriber
2. Subscription
3. Usage: watch, visit, apply, read, …

Product Business (PB)

1. Customer
2. Order
3. Order detail: including all SKU details merchants need to track.

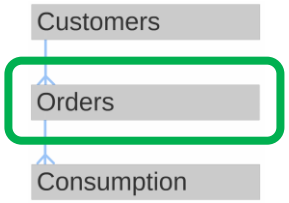
Remember each level is to have “everything there is to know” about the level. So if our business question is about subscribers, we should only need to look at the subscriber table.



SB: 1. Subscriber Summary

“Everything” about a subscriber.

- PK: Subscriber ID
- First Name (only, rest of PII restricted)
- Is Currently Subscribed flag
- Initial & Last Subscription dates & products
- Recency: Last Subscription Chain End Date, Last Usage Stint
- Frequency & Counts: # Chains, # Payments, # Products, # Usage Stints
- Monetary: RTD, RTD 1st Chain, RTD <initial x months>
- Tenure: # Days, # Days Subscribed, % Coverage
- Acquisition Details: Channel, Offer, …
- Consumption Profiles: Initial, Latest, Breadth
- General Engagement: site, email, call center, …
- Demographics: Neighborhood & Individual Append
- Cohort Flags: YYYY, YYQQ, YYMM
- Segments & Scores from data science
- Any alternate Subscriber identifiers (FKs to other systems)
- More ????

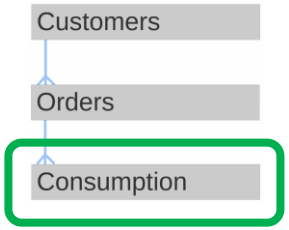


SB: 2. Chained Subscription Summary

“Everything” about a subscription chain.

A subscription chain models the CIA to start a subscription and, optionally, stop or change the subscription.

- PK: Subscriber ID, Chain Sequence #
- Is Subscription Active flag
- Products: This chain, prior chain, next chain
- Dates: Starting & Ending
- Intervals: Days to prior chain. Days to next chain.
- Payments: # and $s
- Initial Conditions: status, promo, offer, …
- Cancel: Requested On, Is Voluntary, Stated Reason
- Consumption within chain: level, type, breadth, acceleration
- More??

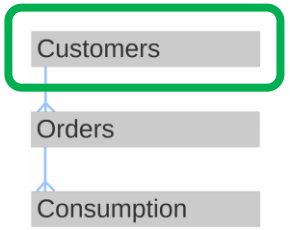


SB: 3. Usage Stint Summary

"Everything" about a usage stint.

A usage stint models the CIA to start continuous consumption and then to stop. For fitness club: entering & leaving. For on-line training: start watching & stop watching for more than 30 minutes.

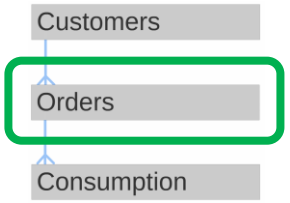
- PK: Subscriber ID, Stint Sequence #. (AK: Chain Sequence #)
- Timestamps: Start & End
- Usage Profile:
  - What used – perhaps multiple factors describing content
  - How many used
  - Where used – location, platform
  - More ???



PB: 1. Customer Summary

“Everything” about an order.

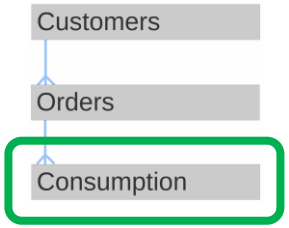
- PK: Customer ID
- First Name (only, rest of PII restricted)
- Status Flags: Is Registered, Has Purchased, Is <special>, …
- Initial & Last Order Dates, Amounts, & Top Product Focus
- Following are perhaps also split out by product group:
  - Recency: Days since last order
  - Frequency & Counts: # Orders, # Product Groups, # SKUs
  - Values: RTD, RTD<initial x months>, Average Order Value
- Interval: Tenure Days (days since first order)
- Order Rates: Orders per [year, quarter, month], frequency variance
- Seasonal & Life-stage purchase RFM’s
- Acquisition Details: Initial Channel, Offer, Initial Product Group, …
- General Engagement: site, email, call center, …
- Demographics: Neighborhood & Individual Append
- Cohort Flags: YYYY, YYQQ, YYMM
- Segments & Scores from data science
- Any alternate Subscriber identifiers (FKs to other systems)
- More ????



PB: 2. Order Summary

“Everything” about an order.

- PK: Order ID; FK: Customer ID
- Order Sequence # (for individual customer)
- Order Date / Time
- Is Most Recent Order flag
- # Line Items
- Intervals: Days to Prior Order, Next Order
- Payments: Total. Perhaps break down: merchandise, shipping, tax, …
- Discounts & Offer codes
- Returns & Refunds
- Source: Channel, Referral, …
- Top Product Group. % Dominance of Top Group.
- Geo location: Billing & Shipping
- Follow-up Satifaction
- More???

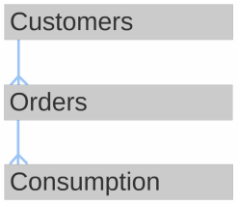


PB: 3. Order Line Detail

"Everything" about an order line.

- PK: Order ID, Line #. FK: Customer ID
- SKU, Description
- Quantity
- SKU specific offer, discount
- Product kind, group, class, … (function of product taxonomy)
- Price(s):
  - Standard
  - Actual
  - Breakdown by components & options
- Primary Attributes: Size, Color, ….
- Attributes of Options: (Function of option)

This can be very complex for SKU's with many options. Merchants will want to slice & dice along option categories.



The Structure in Action

This customer insights focused data structure has two major benefits:

1. Business analysts can easily get consistant data driven answers to most of their customer related questions – *by themselves!*
2. Data scientists have a clean, rich data set for modeling that will sync up with business results.

Following examples will use SQL. The same logic applies to using a visual tool like Tableau.

SQL Ex 1: How many subscribers do we have?

Question is about subscribers. So we use the 1st level of abstraction – Subscriber Summary table. SQL is trivial! Just count the number of subscribers who have Is Subscribed = TRUE.

```
SELECT COUNT(*) AS Number_Subscribed
  FROM subscriber_summary AS ss
 WHERE ss.is_subscribed;
```

This works because the complex business rules defining if someone is subscribed are built into the Is Subscribed flag.

The process of building the Subscriber Summary table is non-trivial to ensure it is "analyst friendly!"

SQL

Ex 2: By monthly cohort, how many, and what % of, paying subscribers are currently subscribed?

Question is also about subscribers. Use the Subscriber Summary table again.

```
SELECT ss.cohort_yymm,
       COUNT(*) AS Number_Starts,
       SUM(ss.is_subscribed::INT) AS Number_Subscribed,
       (100.0 * SUM(ss.is_subscribed::INT) /
         COUNT(*))::NUMERIC(10, 1) AS Percent_Subscribed
  FROM subscriber_summary AS ss
 WHERE ss.revenue_to_date > 0
 GROUP BY 1
 ORDER BY 1;
```

“paying subscribers” have a RTD > $0.
“monthly cohort” held in the Cohort YYMM column. Use in GROUP BY.
COUNT(*) gives # subscribers who have paid, ever.
SUM(…) gives the # who are currently subscribed.

SQL

Ex 3: By cancel reason, what % of 1st chain cancels re-subscribe and what is average lag (2014 onward)?

Question is about subscriptions – 2nd level. Use Subscription Summary.

```
SELECT css.ending_cancel_reason,
       COUNT(*) AS Number_1st_Chain_Cancels,
       COUNT(css.days_until_next_start) AS Number_Resubs,
       (100.0 * COUNT(css.days_until_next_start) /
         COUNT(*))::NUMERIC(10, 1) AS Percent_Resubs,
       AVG(css.days_until_next_start) AS Average_Days_Before_Resub
  FROM chained_subscription_summary AS css
 WHERE css.chain_sequence_number = 1
   AND NOT css.is_active
   AND css.starting_date >= '2014-01-01'
 GROUP BY 1
 ORDER BY COUNT(*) DESC;
```

"1st chain cancels" – Chain Sequence # = 1 & Is Active is not TRUE.
"2014" onward – Starting Date from Jan 1, 2014 to date.
"By cancel reason" – GROUP BY Ending Cancel Reason for chain.
Days Until Next Start will be NULL if no 2nd chain.

SQL Ex 4: What are the top five content areas viewed over the last 90 days?

Question is about usage – 3rd level. Use Content Usage Stint Summary.

```
SELECT cuss.top_area AS Top_Area_In_Stint,
       COUNT(*) AS Number_Stints
  FROM content_usage_stint_summary AS cuss
 WHERE cuss.stint_start_at >= DATEADD(day, -90, GETDATE())
 GROUP BY 1
 ORDER BY 2 DESC
 LIMIT 5;
```

“last 90 days” – usage stint starts after 90 days ago via GETDATE()
“top content areas” – stint’s Top Area – the GROUP BY
but ORDER BY the # stints for the content area in descending order, and
LIMIT to just top 5

Key Takeaways

1. “The (data) structure stupid!”
2. Focus on CIA’s (Customer Initiated Actions)
3. Be analyst ready.
4. You are a (data) publisher! Act like one.
5. Start simple, then expand what “everything” means as you roll out to the business.
6. Make the investment in structured data. It will pay off!

Wrap Up

Questions?
Comments?
Now is the time!

See Jim's archive at DS4CI.org for

- These slides.
- The related White Paper.
- And much more!

Contact me at: Jim@DS4CI.org

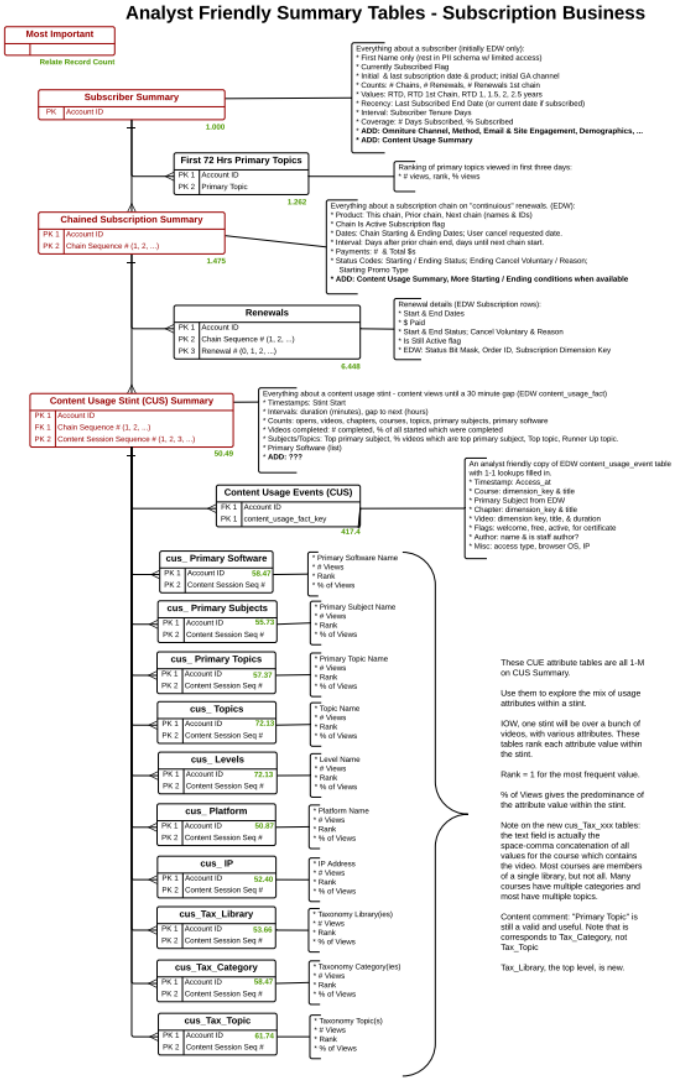
Appendix follows with

- *Real world subscription example*
- *Sessionization in SQL & R*

APPENDIX

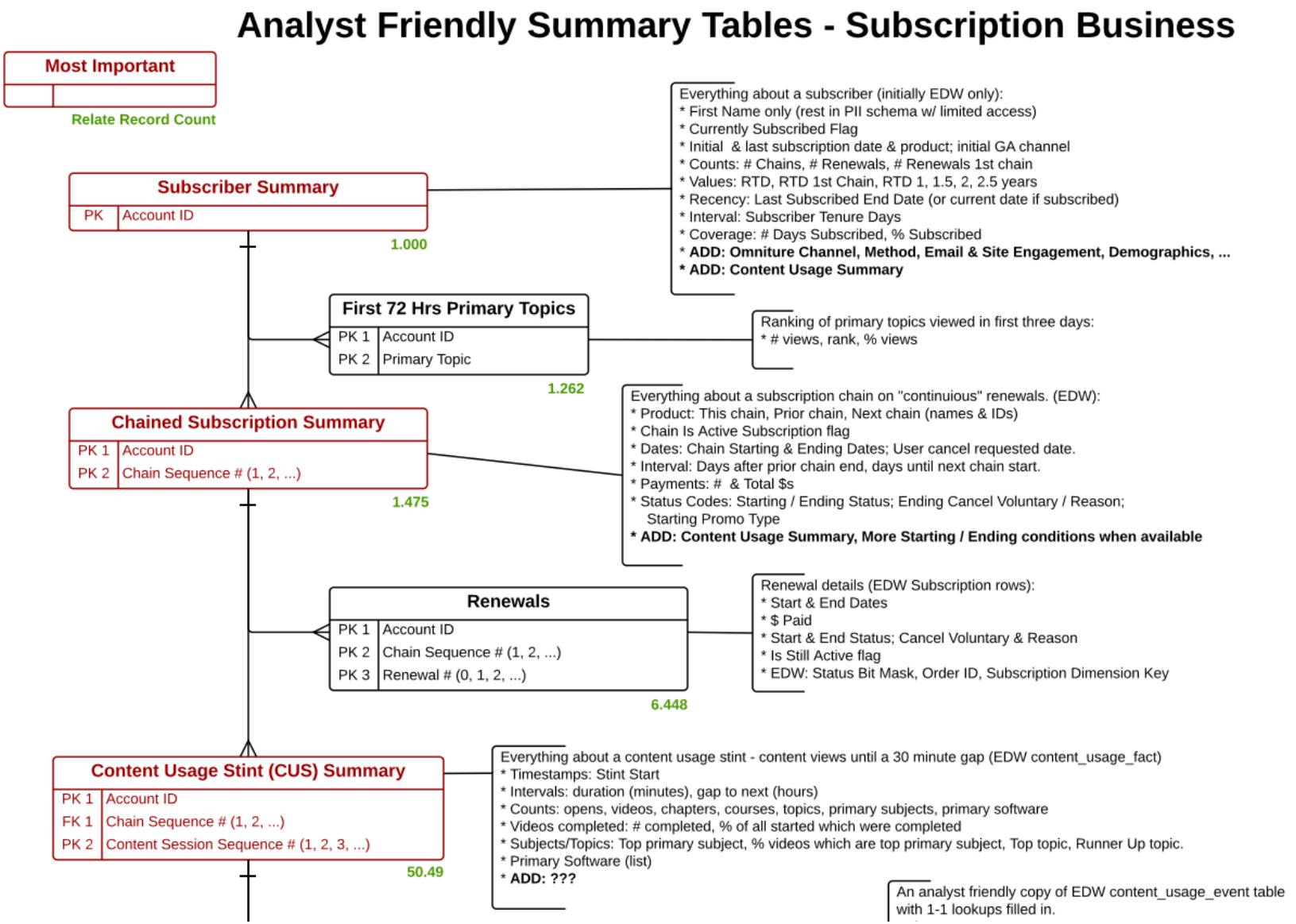
1. Real-world subscription business example.
2. Sessionization in SQL and R

Real-world Subscription Model

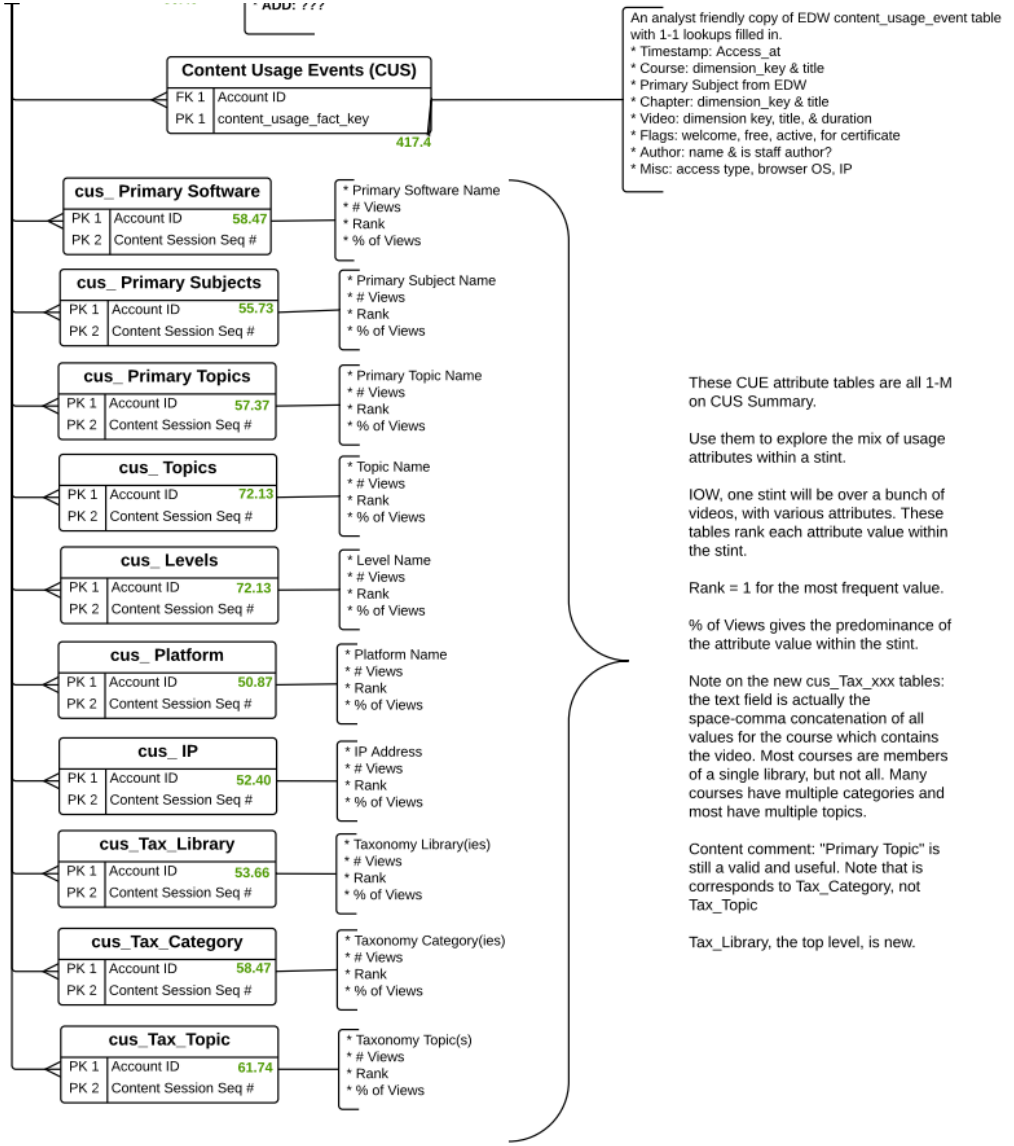


- Red Tables are the three levels of abstraction.
- Green #'s show relative number of records to Subscriber Summary
- Here we just show keys and description of contents.

Real-world Subscription Model - Top



Real-world Subscription Model - Bottom



- Content Usage Events is an analyst friendly version of the video view event fact table w/ all dimension joins done.
- The cus_xxx table rows have counts and % of total for each value in CUS dimensions within a Content Usage Stint
- These fill the Top_xxx and Pct_xxx values in the Content Usage Stint Summary.
- To take a deep dive within the stint look at the cus_xxx tables.

Sessionization Methods

Content Usage Stint Summary and Chained Subscription Summary tables are rollups of event level data. These are special cases of the general event sessionization problem, which can be done in any modern SQL platform or in R (with dplyr package).

The next slides will cover the highlights in both languages.

For background and details see Jim's archives at:

- For SQL see Sessionization Methods – Getting Past Big Data Noise
  - Especially slides 14-17 onwards and 26-28
- For R see dplyr Example 2 – Sessionize Web Events

Sessionization in PostgreSQL (1 of 3)

Problem: Sessionize web events by member ID using the 30 minute end-of-session rule.

```
-- get event sequence #s & seconds after prior event
CREATE TABLE v_sessions1 AS
SELECT *,
       ROW_NUMBER() OVER(Members) AS event_seq_number,
       event_at - LAG(event_at) OVER(Members)
           AS interval_to_prior
  FROM v_events
WINDOW Members
    AS (PARTITION BY member_id    -- unique member ID
            ORDER BY event_at     -- timestamp of event
       )
;
```

Sessionization in PostgreSQL (2 of 3)

```
-- update with session sequence #
CREATE TABLE v_sessions2 AS
SELECT *,
       COUNT(CASE WHEN interval_to_prior IS NULL OR
                       interval_to_prior > '30 minutes'
                  THEN 1 ELSE NULL END) OVER(Members)
           AS session_seq_number
  FROM v_sessions1
WINDOW Members
    AS (PARTITION BY member_id       -- unique member ID
          ORDER BY event_seq_number  -- Session #
       )
;
```

Sessionization in PostgreSQL (3 of 3)

```
-- now roll up into sessions getting session start, total time in session,
-- site areas explored, other site specific rollups

CREATE TABLE v_sessions AS
SELECT member_id,
       session_seq_number,
       MIN(event_at) AS session_start_at,
       COUNT(*) AS num_events_in_session,
       SUM(CASE WHEN interval_to_prior > '30 minutes'
                THEN NULL ELSE interval_to_prior END) AS session_duration,
       STRING_AGG(DISCRETE site_area, ', ') AS site_areas_visited,
       <other site specific aggregations>
  FROM v_sessions2
 GROUP BY member_id,
          session_seq_number
 ORDER BY member_id,
          session_seq_number
;
```

Sessionization in R w/ dplyr

Problem: Sessionize AOL search events by ID using the 30 minute end-of-session rule.

```
aol_sessions <- aol %>%
  arrange(AnonID, QueryTime) %>%
  group_by(AnonID) %>%
  mutate(Minutes_After_Last = difftime(QueryTime, lag(QueryTime), units = "mins"),
         New_Session_Flag = is.na(lag(AnonID)) | Minutes_After_Last > 30,
         Session_Seq_Num = cumsum(New_Session_Flag)
         ) %>%
  group_by(AnonID, Session_Seq_Num) %>%
  summarize(Session_Start_At = first(QueryTime),
            Number_Searches = n(),
            Number_Terms = n_distinct(Query),
            Session_Duration_Minutes = as.numeric(difftime(last(QueryTime), first(QueryTime),
                                                           units = "mins")),
            Number_Clicks = sum(!is.na(ClickURL))
            )
```

For full details see [dplyr Example 2 - Sessionize Web Events]