Overall approach

The ML team will devise different regression models based on “super-categories” (as labeled by Nolan). These are the “invariant”, “seasonally variant”, and “other” categories we have been describing since our first presentation. They cluster nodeIDs by the way the sales of books within that nodeID vary month by month. So all nodeIDs in the “invariant” super-category sell about the same amount of books every month. All nodeIDs in the “seasonally-invariant” super-category sell more books in a one or two month period (e.g. Nov/Dec) than in any other period. Everything in the “other” category is anything that doesn’t fit in the first two.

In our EDA, we have found 4 clusters which we believe will be good supercategories (see our Notebook). We have a SQL script which can find these categories. We just need a place to store them for future use. We request a lookup table in the schema which would allow us to store nodeIDs with their associated supercategory (so a two column table). Ultimately, we’ll need a way to get a list of all nodeIDs in a certain supercategory. Conversely, we’ll need a way to search for the supercategory for a specific nodeID. We assume this will be just a SQL query if the schema has the supercategory lookup table included.

To train the model, we need to query the datasource like this:

For supercategory X, return the data for these fields:

- NodeID
- Year
- Month
- Dollar amount sold last month

- Dollar amount sold last quarter
- Dollar amount sold last year
- Number sold last month
- Number sold last quarter
- Number sold last year
- Average rating for category
- Difference between this category rating and all others
- Sentiment Factor Value
- Is in Campaign?

We've provided an example Notebook to show the basic approach for the model.

Here's the SQL query for a virtual view of books sold per nodeID per month:

```
CREATE VIEW monthly_sales AS
select EXTRACT(MONTH from o.billdate) as mon, p.nodeid as category,
sum(o.numunits) as books_sold
from orderlines as o, products as p
where o.productid = p.productid AND o.totalprice > 0::money
group by p.nodeid, EXTRACT(MONTH from billdate)
order by p.nodeid
```

So if we were asked the top 3 nodeIDs for Christmas we could use this view:

```
SELECT category, sum(books_sold) AS num_sold FROM monthly_sales
WHERE mon = 11 or mon = 12
```

```
GROUP BY category
ORDER BY num_sold DESC
LIMIT 3
```

We'll be using the view of monthly_sales (by volume of books sold and by total sales in dollars) and request that this be allowed by the schema.

We'll also need to use the PARTITION and LAG SQL commands. These aren't strictly defined in Datalog so we'll define them in the same way that we define SUM, COUNT, and other aggregation functions. Nevertheless, the SQL query is easy to write. For example, this query returns the sales trend (positive or negative number of books sold) from month 3 to month 5 for all the nodeIDs (i.e. which books increase their sales in Spring):

```
SELECT s.category, round(avg(s.change_in_sales_from_last_month))
AS sale_trend
FROM
(
SELECT category, mon, count(mon) OVER (PARTITION BY category) as
num_months,
books_sold - lag(books_sold,1) over (PARTITION BY category ORDER
BY mon) as change_in_sales_from_last_month
FROM monthly_sales
WHERE mon >= 2 AND mon <= 5
GROUP BY category, mon, books_sold
) AS s
WHERE s.num_months = (5-2) AND s.mon > 2 AND s.mon <= 5
GROUP BY s.category
ORDER BY sale_trend ASC
```

Specific Queries

1. Dollar amount sold last month
2. Dollar amount sold last quarter
3. Dollar amount sold last year
4. Number sold last month
5. Number sold last quarter
6. Number sold last year
7. Average rating for category
8. Difference between a category rating and all others
9. Sentiment Factor Value
10. Is in Campaign
11. Sum of Books Sold (for an year and NodeID)
12. Super Category
13. What are the top 3 categories of books that are most read around Christmas?
14. What time of the year are the sales of “Education” books the highest?
15. Given month m and category c, predict the amount of sales for the category.
16. Which book categories show a downward trend in demand in Winter and Spring?
17. Is there a category that we should discontinue stocking?

CREATE VIEW MonthlyDollarsSalesView :- (nodeid, month, totalsalesprice)
salesaggregate(nodeid, month, year, totalsalesprice)

CREATE VIEW MonthlyUnitsSalesView :- (month, nodeid, numunits)
salesaggregate (nodeid, month, year, totalsalesvolume),
Products(nodeid)

CREATE VIEW YearlyDollarsSalesView :- (nodeid, year, totalsalesprice)
salesaggregate(nodeid, year, totalsalesprice)

CREATE VIEW YearlyUnitsSalesView :- (year, nodeid, numunits)
salesaggregate (billdate, numUnits, nodeID),
Products(nodeid)

Q1(category, totalsalesprice) :- salesaggregate(nodeid, month, year,
totalsalesprice),
month = month - 1
Q2(category, totalsalesprice) :- salesaggregate(nodeid, month, year,
totalsalesprice)
month = month - 3
Q3(category, totalsalesprice) :- salesaggregate(nodeid, month, year,
totalsalesprice),
year = year - 1
Q4(category, totalsalesvolume) :- salesaggregate(nodeid, month, year,
totalsalesvolume),
month = month - 1
Q5(category, totalsalesvolume) :- salesaggregate(nodeid, month, year,
totalsalesvolume),
month = month - 3
Q6(category, totalsalesvolume) := salesaggregate(nodeid, month, year,
totalsalesvolume),
year = year – 1

Q7 (categoryid, avgrating) := sum(reviews(asin, avgrating),
sales(asin, categoryid))

Q9 (SentimentFactor) := (nodeid)
Q10
Q11(category, totalsalesvolume) :- sum(
(salesaggregate(nodeid, month = 1, year, totalsalesvolume)
(salesaggregate(nodeid, month = 2, year, totalsalesvolume)
(salesaggregate(nodeid, month = 3, year, totalsalesvolume)
(salesaggregate(nodeid, month = 4, year, totalsalesvolume)
(salesaggregate(nodeid, month = 5, year, totalsalesvolume)
(salesaggregate(nodeid, month = 6, year, totalsalesvolume)
(salesaggregate(nodeid, month = 7, year, totalsalesvolume)
(salesaggregate(nodeid, month = 8, year, totalsalesvolume)
(salesaggregate(nodeid, month = 9, year, totalsalesvolume)
(salesaggregate(nodeid, month = 10, year, totalsalesvolume)
(salesaggregate(nodeid, month = 11, year, totalsalesvolume)
(salesaggregate(nodeid, month = 12, year, totalsalesvolume)
)

Q12 > TBD. An attribute need to be created for each of the top 75 categories based upon the analysis from the EDA team.

We have identified the top 75 nodeIDs:

```
CREATE VIEW top75 AS
SELECT nodeID FROM monthly_sales GROUP BY nodeID ORDER BY sum(books_sold)
DESC LIMIT 75
```

We request that this virtual view be made a part of the schema so that we can query it. It might be nice if the number (75) were made a variable so that we can query the top 10 or the top 200 with the same command.

Q13(category, books_sold) := sum(MonthlyUnitsSalesView(books_sold), month = 1:12,
GROUP BY category

ORDER BY num_sold DESC
limit 3

Based on to Q15 ML model with a specific set of NodeId's e.g. (matchin Education)

Q14() & Q15

This is where the ML model will predict, # require attribute

Q16
For reference, working sql for winter & spring trend is below, which can be used to directly create the schema queries (without going through the Datalog queries owing to complex “Partition” and “Lag” function usage from Postgres)

```
-- Spring
SELECT s.category, round(avg(s.change_in_sales_from_last_month)) AS sale_trend
FROM
(
SELECT category, mon, count(mon) OVER (PARTITION BY category) as num_months,
books_sold - lag(books_sold,1) over (PARTITION BY category ORDER BY mon) as
change_in_sales_from_last_month
FROM monthly_sales
WHERE mon >= 2 AND mon <= 5
GROUP BY category, mon, books_sold
) AS s
WHERE s.num_months = (5-2) AND s.mon > 2 AND s.mon <= 5
GROUP BY s.category
ORDER BY sale_trend ASC

-- Winter
SELECT s.category, round(avg(s.change_in_sales_from_last_month)) AS sale_trend
FROM
(
SELECT category, mon, count(mon) OVER (PARTITION BY category) as num_months,
books_sold - lag(books_sold,1) over (PARTITION BY category ORDER BY mon) as
change_in_sales_from_last_month
FROM monthly_sales
WHERE mon >= 8 AND mon <= 12
```

```
GROUP BY category, mon, books_sold
) AS s
WHERE s.num_months = (12-8) AND s.mon > 8 AND s.mon <= 12
GROUP BY s.category
ORDER BY sale_trend ASC
```

Following are the DataLog queries for Winter and Spring trends

```
CREATE VIEW Winter_Trends (category, mon, count(mon) OVER (PARTITION BY
category) as num_months) :=
                        MonthlyUnitsSalesView(numunits - Lag(numunits, 1) over
                         (PARTITION BY category ORDER BY mon)
                          as change_in_sales_from_last_month )
                         (mon >= 8 AND mon <=12)


  Q16_Winter(category, (change_in_sales_from_last_month))) : =
        Winter_Trends (round(avg (change_in_sales_from_last_month ) )
        Group By category
        Order By change_in_sales_from_last_month ASC


CREATE VIEW Spring_Trends- (category, mon, count(mon) OVER (PARTITION BY
category) as num_months) :=
                        MonthlyUnitsSalesView(numunits - Lag(numunits, 1) over
                        (PARTITION BY category ORDER BY mon)
                         as change_in_sales_from_last_month )
                         (mon >= 2 AND mon <=5)

  Q16_Spring(category, (change_in_sales_from_last_month))) : =
         Spring_Trends (round(avg (change_in_sales_from_last_month ) )
        Group By category
        Order By change_in_sales_from_last_month ASC

Q17(category) := YearlyUnitsSalesView(category)
                                sum(numunits) < 4
                                 (yr < 2017 AND yr > 2013)
```