DATA STRATEGY & LOOKER HEALTH CHECK

Sam Padgett

Table of Contents

Project Details

Project Name	Data Strategy - SOW #1
Project Type	Assessment & Data Strategy
Project Start Date	8/8/22
Project End Date	9/13/22
Project Sponsor	
Project Manager	

Executive Overview

Pander conducted pre-engagement interviews and surfaced the concerns and resulting impacts captured in the table below:

Concern	***Desired Impact***
Concern around best practices on current Looker implementation and consistency in LookML	Ability to evaluate and implement the best performance out of the current tech stack - Structure and boundaries for each tool (dbt & Looker) Ability to evaluate the advantages of utilizing Looker with BigQuery vs Redshift
Current pain points around batch processing and refresh times ***Goal*** is to be closer to realtime, especially around the holidays	Ability to evaluate the best utilization of Look ML vs dbt - Improve best practices and consistency

Consistency of computed metrics within Looker	Ability to explore the discoverability within Looker

Client has a modern cloud based ELT architecture consisting of DBT for the ELT layer and Looker for reporting. Our assessment focused on utilization of these components from a best practices perspective.

We also provided guidance on a possible migration to BigQuery to overcome constraints in the current Redshift database platform and leverage BQ Machine Learning functionality.

We conducted thirteen collaborative workshops where we investigated the current dbt based data pipeline, demonstrated methods, techniques, and third party tools for analyzing Looker and managing change across the platform. We also demonstrated the integration of BigQueryML within the Looker environment, and reviewed "beyond vanilla BI" capabilities and resources in the context of a "Looker - Art of the Possible" presentation.

In lieu of a POC deliverable, we spent several collaborative sessions working through a Looker change management approach for swapping out references to the retired tagging functionality in Zendesk.

High level recommendations include:

1. Migration of derived table logic to dbt
2. Modularization of the ELT pipeline
3. Looker environment cleanup guided by the output of the demonstrated audit tools and methodologies.
4. Creation of LookML versions of dashboards for enhanced change management and inclusion in source code control.

Additional details and best practices guidance are included in the content below and supplemented by the provided attachments.

Current State Assessment

The process of investigating the concerns raised in the pre-engagement interviews included a review of Client's current environment focusing on the following areas:

- Model Review
- Evaluate dbt pipelines

- Evaluate LookML layer

Model Review

Client's application model is built on the synergistic capabilities of dbt and Looker. While extremely complimentary, overlaps in functionality make the decision on where to draw the boundary between the two tools confusing.

At Contractor, we're seeing our larger customers move sql application logic formerly housed in Looker derived tables to dbt. This has the following benefits:

- Ability to leverage the dbt's lineage graph feature.
- PDT's - materialization management consistent with existing pipeline. An incremental update option is available within Looker, but dbt is the prefered choice.
- Subsequent modularization, optimization, dependency management. (Presumes an initial 1:1 migration to dbt. Modularization is discussed below.)

Looker supports two varieties of derived tables - Native, based on the LookML modeling language; and SQL-based, as the name implies, built on a sql script. Both varieties have the capability to be persisted (i.e. materialized).

The current count of derived tables across the Client project are shown in the table below.

	PDT	Non PDT
Native derived table	3	24
SQL-based derived table	16	79

The color intensity indicates suggested priority for movement to dbt - darkest first,etc..
As a general strategy, migration is based on the following:

PDT's first - scheduling, incremental updates
SQL-based - More complex business logic here
Non PDT/NDT - low priority - typically simple aggregations

Any tables that have dynamic filters (i.e. filters based on user supplied prompt responses), can not be moved to dbt. PDT's inherently do not support dynamic filters, and interestingly, there are no dynamic filters on any of the other derived tables.

The reference material includes a python script we used to extract the sql from the SQL-based derived tables and construct ‘legacy’ models for the derived tables in the KPI Views folder.

Evaluation Approach

The pipeline and Looker audit was conducted in conjunction with eleven collaborative work sessions equally split across the two subject areas.

Two additional sessions were conducted to demonstrate:

- Looker hosted BigQuery Machine Learning prototype
- Discuss BigQuery Migration Workstreams and SQL Translation Aids

These work sessions are enumerated in attachment I, along with links to produced collateral and/or public reference materials.

Pipeline Evaluation

Client’s dbt instance primarily serves as a repository for the raw data and base tables used to create derived tables in their Looker instance. In order to provide a more focused and deeper understanding of the current pipeline, the queries used to create Looker’s KPI views were selected for analysis as the KPIs are priority metrics used for top end reporting.

The analysis process was carried out by creating flowcharts for each KPI table and a detailed source to target breakdown to determine tables pulling from raw data rather than dbt sources as well as helping to identify overlapping joins and/or duplication of effort amongst the tables that create the kpi tables.

- Source-to-target breakdown

	A	B	C
1	kpi_base	level 1	level 2
39	kpi_b2b_pipeline	b2b_orders_xf	source('useful_lists','fiscal_calendar')
40	kpi_bing	bing_campaign_performance	bing_ads.campaign_performance_report
41	kpi_components	order_level_build	oms_suborder_calculations
42	kpi_components	order_level_build	oms_suborder_calculations

- Derived table breakdown

		explore: kpi_base {
		view_name: kpi_orders
		view_label: "Orders"
join: kpi_refunds {	pdt	join: kpi_refunds {
		type: left_outer
		view_label: "Braintree Refunds"
		relationship: one_to_one
		sql_on: ${kpi_orders.delivery_week} = ${kpi_refunds.refund_week}
		AND ${kpi_orders.distribution_point_city} = ${kpi_refunds.distribution_point_city}
		AND ${kpi_orders.is_subscription} = ${kpi_refunds.is_subscription};;
		}

-

The findings of the analysis are as follows:

- Many of the initial steps to create a more efficient and mature data stack are already present: table organization, complex queries in dbt, documentation
- Only a small portion of Looker views are tied to directly to corresponding dbt tables (dimension and fact tables)
- Dbt sources aren't clearly defined and in many cases tables are pulling from raw data sources
- Tables are created in a somewhat ad hoc nature rather than following a formal structure or built with modularity in mind (dbt modularity discussed further below)
- There is some testing within the dbt tables, but no formal governance
- Dbt tables are version controlled in github, but no formal code review or approval needed before deploying a PR. (This situation is organic as there is only one dev currently working on the dbt and Looker pipeline, but a formal process should be implemented as new devs are added to the team)

Looker Instance Evaluation

Tasks, techniques, and tips on change management tasks covered in the course of our evaluation are enumerated below.

A. Object Inventory

Initial evaluation steps were to construct an object inventory across the Looker instance summarized below with details referenced in the attached.

High Level Object Inventory

Derived (SQL) Table	50

DB Table/View	112
Derived (Native) Table	26
Explores	85
Models	6

The Inventory level analysis was enhanced via the use of Henry, a third party utility that surfaces unused (in last 90 days) models, explores, joins and fields.
Insert table

Example Results

Model	Explore	Fields	Unused Fields
Topline Sales	sales_date	1991	1080

OUr analysis was augmented through the use of Looker's search objects feature. Various search criteria identified in attachment IV.

B. System Administrator Review

Highlights from Looker's System Activity reporting are captured below

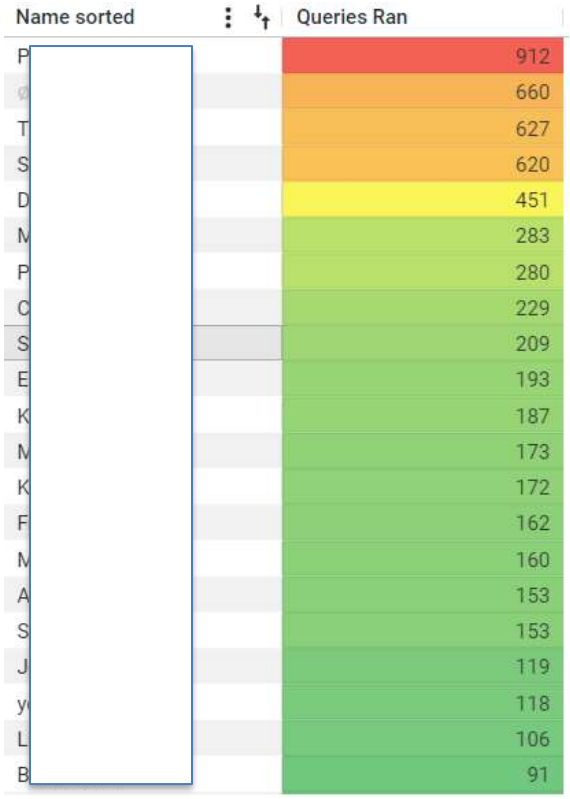
1. User Activity

User Accounts & Adoption

Total Users	75
Standar Users	23
View only Users	46
Weekly Avg Querying Users	49
Percent of users active last 7 days	55%

Total Users	75
Standar Users	23
Avg Minutes per User	75
Avg Queries per User	150

Top Users



Top Sources

Source sorted	Queries Ran
dashboard	4125
explore	704
regenerator	660
look	612
suggest	195
render_manager_precache:dashboard	102
sqlrunner	86
data-download-api	76
render_manager_cache:dashboard	57
alerts	14
scheduled_task	10
private_embed	9

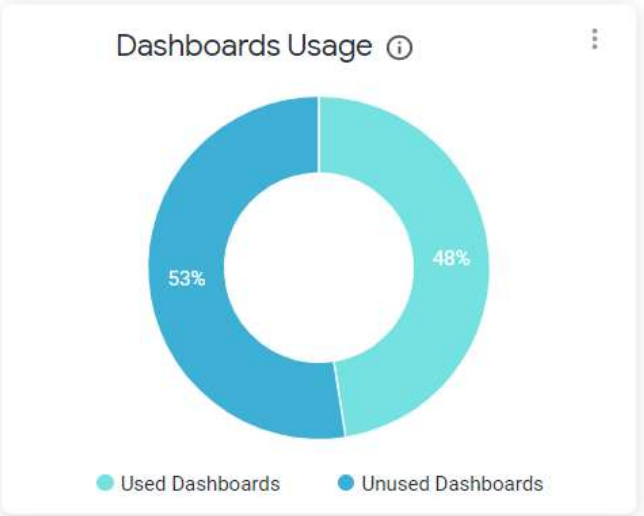
The low values for alerts and scheduled tasks suggests possible opportunities to increase utilization of these features.

2. Content Activity

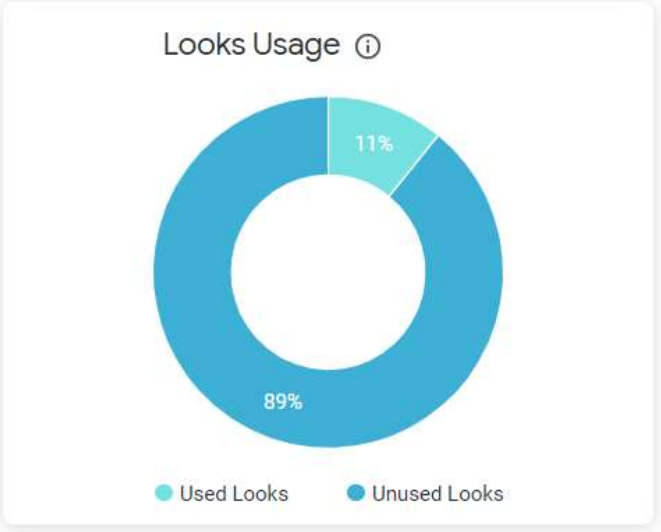
User Accounts & Adoption

Total Dashboards	160
Total Looks	1295
Scheduled Plans	29
Schedule Distribution	good

Dashboard Usage



Looks Usage



3. Database Performance (Last 7 days)

Results from cache

The results from cache show significant drop from prior period, which could be the result of our activity in the instance. This should be re-examined in the coming weeks. Replication of the recommendations in the example dashboard review (specifically dynamic vs static filters) could improve this number in some situations.

Top Explores

Top Explores

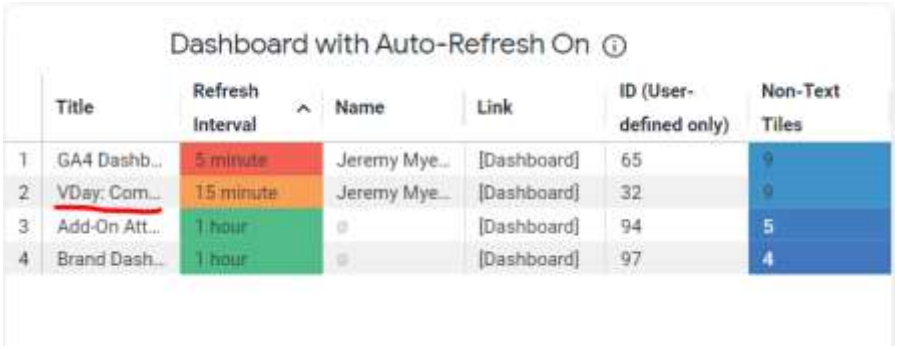
	Explore	Model	Query Run Count	User Count
1	sales_data	ToplineSales	1627	27
2	inventory_data	ToplineSales	390	14
3	component_data	ToplineSales	251	9
4	subscriptions	ToplineSales	151	5
5	kpi_b2b_pipeline	kpis	107	3

Other observations:

- Components_economics pdt build is consistently failing
- At 90 seconds, the segment_flow_custom_query explore has the highest run time by a substantial margin.
- OKR Dashboard, Rolling 12-week KPIs, and QA meeting Dashboard have >9 distinct queries exceeding 30 seconds. These dashboards exceed the 25 tile suggested limit and should be evaluated for possible "modularization".

4. Instance Performance

Note - Valentine Day Dashboard being refreshed every 15 minutes.



Dashboard with Auto-Refresh On

	Title	Refresh Interval	Name	Link	ID (User-defined only)	Non-Text Tiles
1	GA4 Dashb...	5 minute	Jeremy Mye...	[Dashboard]	65	9
2	VDay: Com...	15 minute	Jeremy Mye...	[Dashboard]	32	9
3	Add-On Att...	1 hour		[Dashboard]	94	5
4	Brand Dash...	1 hour		[Dashboard]	97	4

Dashboards with more than 25 tiles

Title	Link	Total Query Tiles	Total Queries Generated
Rolling 12-week KPIs ...	[Dashboard]	35	35
Rolling 12-week KPIs	[Dashboard]	35	35
Rolling N-week KPIs	[Dashboard]	33	33
FY 22 OKR/ KPI Report	[Dashboard]	31	30
Gender Pre-Ordering	[Dashboard]	26	26

C. Content Validation Output Review

Looker's Content Validation tool identifies errors within the application by object type (Dashboard or Look), Content Name, Folder, Model, and Error detail. To facilitate priority setting, aggregation, analysis, and progress tracking as the error backlog is dealt with, we've constructed a spreadsheet capturing current state, with tips on its production here.

High level summary shows:

Objects with Errors (84 Looks/ 8 Dashboards	92
Individual errors across following error categories:	141
Field not found	114
Unknown Explore	21
Unknown Model	5
Calc references field not in query	1

Note: The System Activity/Errors and Broken Content section provides additional insight. Particularly useful are the impacted user counts and a view of broken Looks by user.

D. Dashboards - User Defined vs LookML

Dashboards constructed in the Looker GUI, are considered 'User Defined'. As we demonstrated in our LookML work session, these dashboards can be converted to 'lookml code based dashboards' by generation of the LookML through the dashboard interface, and pasting to a new (LookML) dashboard file.

100% of Client's dashboards are currently constructed as user defined.

From a user interaction and performance standpoint, there is no difference between these two types of dashboard. However, migrating to LookML brings the dashboard content into the git repository's version management, and content becomes accessible to Looker's search utility..

For example, in the System Administration review, we recommend replacing a static date filter with a dynamic date filter to increase the caching utility caching on a specific dashboard. Once dashboard's are converted (or at least copied) to LookML, a comprehensive search and analysis on 'Filters:' could be done across all dashboard objects. Similarly, dependencies on models, explores, fields, and measures could be mapped.

Absent this step, the referenced dependencies are only accessible via the Looker's Content Validator, which only reports on broken objects.

E. Platform Migration Tips

We discussed platform migration, parallel workstream considerations, Google's batch SQL translation tool, and referenced their Redshift/Bigquery translation guide. See Attachment I for resource links.

Migration of pdt's to dbt provides the advantage of working through sql translation tasks in a single interface for most of the work. However, just a reminder that the remaining LookML still has platform specific syntax in the remaining 'sql:' statements that will need validation.

F. Looker/BQML Integration Demo

Contractor's Joseph L. joined us in one of our work sessions to demonstrate a prototype he had developed leveraging two different BQML models inside of Looker. His code isn't available in a public repository yet, but I've included screenshots from a retail demo with repository links in attachment V.

Future State Recommendations

DBT Pipelines

Recommendations:

A. Move remaining sql based derived tables in Looker to dbt, prioritizing the existing persistent derived tables (PDTs):

	PDT	Non PDT
Native derived table	3	24
SQL-based derived table	16	79

Housing the derived table logic in dbt offers the following advantages:

1. Allows dbt to focus on cleaning, structuring, and validating the tables and Looker displaying them, resulting in more snappy and responsive Looks and Dashboards.

2. Access to dbt data lineage graph.

3. Provides another layer of testing and validation before committing to Looker's production branch.
4. For materialized views, dbt's incremental update option is preferred over Lookers capability. Further, this feature is already in use for existing the pipeline, and avoids introducing complexity into the tech stack.
    a. Dbt incremental table materialization

```
{{
    config(
        materialized = 'incremental',
        unique_key = 'id',
        sort = 'id'
    )
}}
```

We worked with Steven on several examples through an update to the kpi_unified aggregate table that can be used as a template going forward.

As a further assist, we've provided a python script to automate the dbt model construction for this 'legacy' port of the 95 SQL-based views from LookML to dbt. See code comments in the script. The 16 PDT views in this subset will need to be configured for materialization.

B. Dbt Modularity

Dbt modularity is a system of naming, structuring, and relating the tables in dbt to logically and efficiently organize a data stack. This results in a more transparent and standardized data structure that can be easily explained to new developers and others outside of the team. See this dbt modularity document for more information.

Creating a modular data stack can result in a drastic restructuring of the current data tables, so the first step in creating dbt modularity is to migrate legacy code to its own

space within dbt for the system to continue to pull from while the modularity work progresses in the following steps:

1. Create a system of data governance. (dbt and looker data governance demo)
2. Define internal sql standards and formatting. (dbt style guide)
3. Define raw data as dbt sources.
    a. Raw data source

```
    topvsother,
    userlistid
    from google_ads_v1.click_performance_report
```

    b. Dbt source

```
with
data_source as (
  select * from {{source('historical_uploads','b2b_sales')}}
)
```

4. Create modular tables in a staging, intermediate, dimension, and fact hierarchy (call the data sources from the previous step in the staging tables).
    a. Non-modularity

    b. Modularity

5. Restructure tables in small pieces over time removing bloat and overlap similar in the way we worked through them together in co-dev sessions. (Henry)
    a. Kpi_unified table only pulling data from needed reference tables

kpi_unified.sql

models > pandera_paradigm > mart > KPIs > fact > kpi_unified.sql

```
{{
    config(
        materialized = 'table'
    )
}}


with kpi_orders as (
  select * from {{ref('kpi_orders')}}
),

kpi_refunds as (
  select * from {{ref('kpi_refunds')}}
),

kpi_product_cost AS (
  select * from {{ref('kpi_product_cost')}}
),

kpi_fedex_confirmed as (
  select * from {{ref('kpi_fedex_confirmed')}}
),
```

6. Rename columns/dimensions so that there is consistency between naming conventions within the dbt tables and their corresponding Looker views. (audit_helper)
    a. Best practice is to have the dimension names in a Looker view match what is being output by the corresponding dbt file.
7. Increase the documentation within dbt: update dbt yaml files with description of tables, primary keys, and unique dimensions, testing within yaml files.
    a. Yaml without documentation

```
models:
  - name: dim_line_item_union
    columns:
    - name: id
```

    b. Yaml with documentation

```yaml
models:
  - name: dim_line_item_union
    description: union of line item dimensions
    columns:
    - name: id
      description: primary Key
      tests:
      - not_null
      - unique
```

8. Audit the new tables versus legacy tables. (Spectacles, audit_helper)

We introduced three tools during the workshops to help with audit process that will be necessary to create the desired future state for the dbt and Looker instance: audit_helper, Spectacles, and Henry

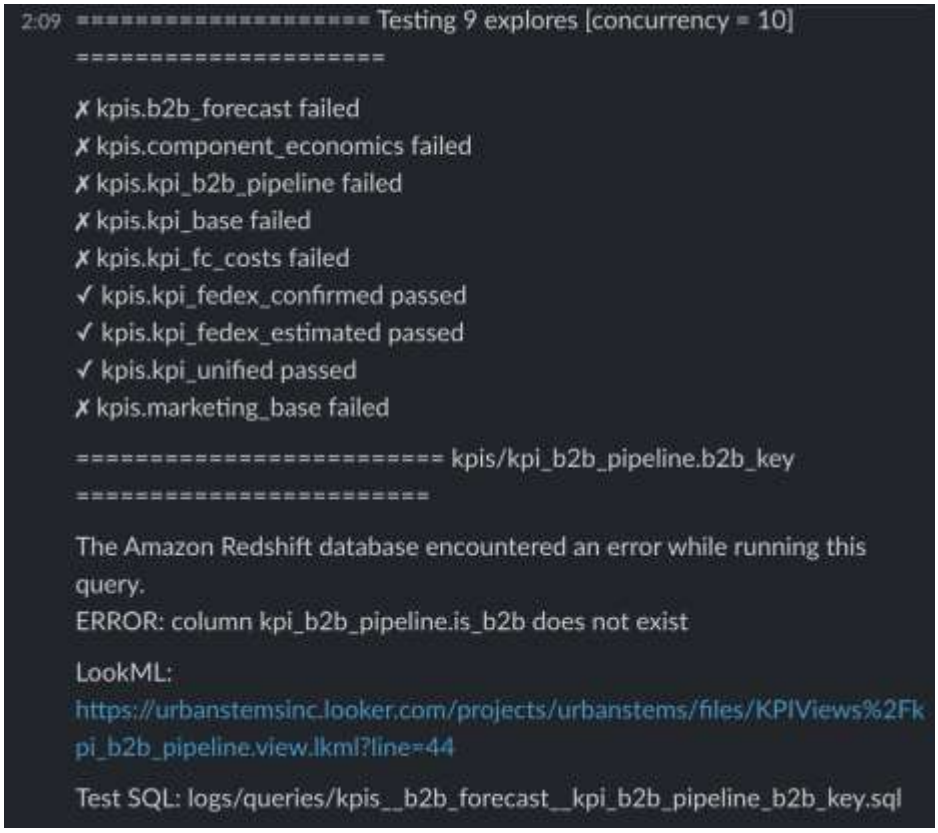
1. audit_helper is a dbt macro, that once the package has been installed, allows the user to conduct a variety of actions to compare tables including:
    a. Row-by-row
    b. Exact column values
    c. Column position and data type (example below)

```
{% set old_etl_relation=adapter.get_relation(
      database='analytics',
      schema='dbt_mgreen',
      identifier='legacy_kpi_fedex_confirmed'
) -%}

{% set dbt_relation=ref('kpi_fedex_confirmed') %}

{{ audit_helper.compare_relation_columns(
    a_relation=old_etl_relation,
    b_relation=dbt_relation
) }}
```

0.3 sec — Returned 8 rows. Download CSV

column_name	a_ordinal_position	b_ordinal_position	a_data_type	b_data_type	has_ordinal_position_match	has_data_type_match
primary_key	1	1	text	character v...	true	false
shipment_we...	2	2	timestamp w...	timestamp w...	true	true
distributio...	3	3	character v...	character v...	true	true
inbound_shi...	4	4	numeric	numeric	true	true
outbound_sh...	5	5	numeric	numeric	true	true
ipd_shippin...	6	6	numeric	numeric	true	true
shipping_co...	7	7	numeric	numeric	true	true
packages_de...	8	8	bigint	bigint	true	true

2. Spectacles is a command line tool (for this project) that can validate lookml code that is currently in production, or code specific to another branch. It has many useful validation functions that are covered in its develop documentation, but the most useful for this instance was using the sql validator. This validator allows the user to spot any errors that the lookml generated sql in Looker's queries may produce before the queries are actually ran in an explore, which can save a lot of time by removing manual steps from the sql validation.
    a. Spectacles sql validation example



3. Henry is a command line tool that helps determine model bloat in your Looker instance and identify unused content in models and explores. It is meant to help developers cleanup models from unused explores and explores from unused joins and fields, as well as maintain a healthy and user-friendly instance. Henry could be particularly useful helping to trim quite excessive bloat found in some of the joins within the tables when making the transition to dbt handling the majority of complex queries.
    a. Henry unused joins and fields output example

Explore	Is Hidden	Has Description	# Joins	# Unused Joins	# Fields	# Unused Fields
netsuite_report	FALSE	FALSE	3	1	180	177
component_data	FALSE	FALSE	11	2	790	728
refund_data	FALSE	FALSE	3	3	44	44
braintree_refund_da	FALSE	FALSE	8	2	164	136
braintree_refund_da	FALSE	FALSE	1	1	5	5
inventory_transactio	FALSE	FALSE	9	6	263	230

Looker /LookML

Recommendations:

A. Application Governance

Implementation of the subsequent recommendations across the breadth of the Client's application is going to require substantial communication with the user community. If not already in place, we highly recommend the cultivation of "product owners" across logical subsets of the application. The following breakout of shared dashboards and looks by folder area is provided as a starting point for reviewing current state of 'lead users' across the organization.

Do not treat this "ownership" designation lightly. Get senior management buy-in to the importance of this role, and celebrate those contributing to the success and expanding impact of the Looker application.

Parent Folder	**Dashboard Count**	**Look Count**	**Owner**
Adhoc		12	BI ?
Admin		13	BI
Archive	2	19	BI ?
B2B	2	18	
Care	12	75	
Care Migration		2	
Customer Experience	2	19	

E-Commerce	3	34	
Examples	8	138	BI ?
Finance		1	Finance ?
KPI's	3	43	Finance ?
Inventory	1	9	
Loyalty Programs		2	Marketing ?
Marketing	7	39	Marketing ?
Cloud Flare		1	NA
Del Reports		1	NA
Totals	**42**	**426**	

B. Retire unused objects surfaced via Henry analysis.

   Other than slowing the validation of LookML changes during development/maintenance, the presence of unused objects does not impact the performance of the Looker instance. However, the same can't be said for the upstream pipeline.

C. Systematically Clear Content Validation Errors

   As we reviewed in one of our workshops, "field not found" errors do not show in Looks or Dashboards. The missing field is simply omitted from the output producing a more aggregated result than originally intended. The error warning is only visible when viewing at the explore level. Consequently, the presence of 114 "field not found errors" is cause for concern.

   Looker's Content Validator is a powerful tool for validating changes, and one of the only options for impact analysis as you pursue retiring unused objects. Weeding through the existing catalog of errors greatly diminishes its utility.

D. Migrate Derived Table logic to dbt

   At the completion of the dbt portion of the derived table migration, create a feature branch and implement the Looker side of the derived table migration. The change

requires deletion of the select statements and replacing with a reference to the new model/view created in dbt.

If the existing names are preserved on the new dbt objects, a script could be constructed to eliminate the tedium of doing this across the 95 derived tables views.

LookML validation and the Spectacles utility should greatly streamline validation of these changes against production.

E. Clear unused Explores / Populate Explore Description Fields for Remaining

Of the 70 explores in the ToplineSales model, the Henry analysis identified 30 as being unused in the past 90 days. These need to be evaluated for removal (recommend use of Content Validator for confirmation of no dependencies - i.e. comment out in model file and execute Content Validator).

Additionally, recommend that explore description fields be populated for the remaining explores. These descriptions are visible to users when they hover over the explore name.

F. Enhanced User Facing Documentation
    a. Population of field descriptions for dashboard measures and ambiguous dimension names. This enable users to click the information icon and see calculation logic, etc...n

G. Increased visibility of support level documentation

Observed several instances of 'point in time' hard coding of filters in some of the view files. For example, Mothers day 2021, Valentines day 201xx. Suggest capturing update requirements and upstream data providers in the project manifest file.

H. Create LookML versions of Dashboards

As discussed in the evaluation section.

I. Scheduling/Caching Review

Datagroups review in conjunction with pdt migration (example - noticed 2021 vday report is on refresh schedule - suspect no ones using?

J. Dashboard Tuning

   i. Tile limits - “modularize” with text box links to facilitate workflow between component dashboards
   ii. Dynamic filters for scheduled dashboards
   iii. Evaluate caching / datagroup assignments for appropriate refresh frequency in conjunction with scheduling
   b. Tighten include statements i.e. */

K. Best Practices/UAT Checklist

L. Derived Table Dynamic Filters Observation
   a. Not using Liquid at all in derived view construction
      i. What does that mean??
         1. Bringing full result set back and filtering on final pass?

M. Prepare for retirement of Enabled Legacy Features
   a. Revert to Legacy Dashboards
   b. Allow double click to select text in textarea in Table Visualizations
   c. Use Legacy LookML Runtime

N. Consider leveraging of Looker’s Homepage, push notifications, and ability to add custom content to the help menu to facilitate user communication. See attachment IV for reference links.

Attachments

Attachment I - Workshops & Collateral

Note: Most of content shared below is housed currently housed on Contractor External drive

	Workshop Topic dbt/pipeline Looker	Date	Notes/References
1	Current State Assessment (challenges, constraints, etc)	8/9	Agreement on pipeline review.
2	Current State Assessment - BI & Reporting Ecosystem Overview	8/10	- LookML analysis and audit
3	Current State Assessment - Data Sources Overview	8/10	Dbt pipeline working session
4	Current State Assessment - Data Storage Overview Redshift	8/11	DBT pipeline working session
5	Current State Assessment - Data Pipelines Overview	8/11	DBT pipeline working session, - dbt- Refactoring SQL for Modularity
6	Current State Assessment - SDLC Overview Evaluate DRY (Don't Repeat Yourself) Coding Best Practices	8/12	- LookML Best Practices
7	Platform Migration - Workstreams/Sql Translation	8/17	- BQ Conversion Workstreams - Batch SQL Migration Tool
8	KPI Unified - pipeline review dbt working session	8/18	- Flowchart of data flows for KPI views - kpi base - target-to-source
9	Looker - “Art of the Possible” deck, and BigQuery Machine Language - Looker demo	8/26	- Looker - Art of the Possible *Deck, demo code repository
10	Looker IDE, Content Validator, UAT Checklist,System Activity Reports / Long running dashboard sample investigation, Review use of Henry utility Added link for python script for legacy table construction in dbt (for derived table views).	8/31	- Looker Analysis with Henry - Looker User Acceptance Testing Guide - Looker UAT Checklist FYI - Looker IDE Tutorial videos: - LookML Editor_1, - LookML Editor_2

			Python script to extract derived table logic to .sql file
11	Workshop - Spectacles - Looker sql validation, etc...	9/2	- Spectacles - Getting Started
12	Spectacles & Zendesk Data Source Change Strategy	9/6	
13	Looker working session - Zendesk Changes (continuation)	9/7	

Attachment II - LookML Work Session Notes

- Misc Looker IDE topics:
  - We highlighted features of Looker's IDE development interface including error detection, context sensitive help reference, and syntax validation.
  - We confirmed Steven's comfort level with Looker's version control capabilities and the git repository configuration.
  - We discussed environment management and use of non-public folders for Dashboard/Look break/fix activity.

- Reviewed Looker's Content Validator and limitations therein.
  - Find/Replace discussion re. Looker's lack of "where used" in look/tile/dashboard functionality and...
    - Hack for "where used"
      1. "breaking" lookml object by commenting out
      2. Execute Content Validator, noting impacted objects
    - Note on field-not-found error
      - Tiles continue to work, just aggregated (missing field removed)
      - Won't see error msg in dashboard. From impacted tiles & looks /explore from here option, will see "field not found, ignored" warning.

- Introduced UAT Checklist and Checklist Guide
  - Not entirely applicable to current engagement, but valuable content as you continue to grow and mature your environment
  - We'll be incorporating some of the items identified here as wrap up review next week.

- Reviewed example of long running dashboard surfaced via Looker's System Activity/Database Performance reporting
  - Split up 33 visual dashboard (25 Looker recommended max).

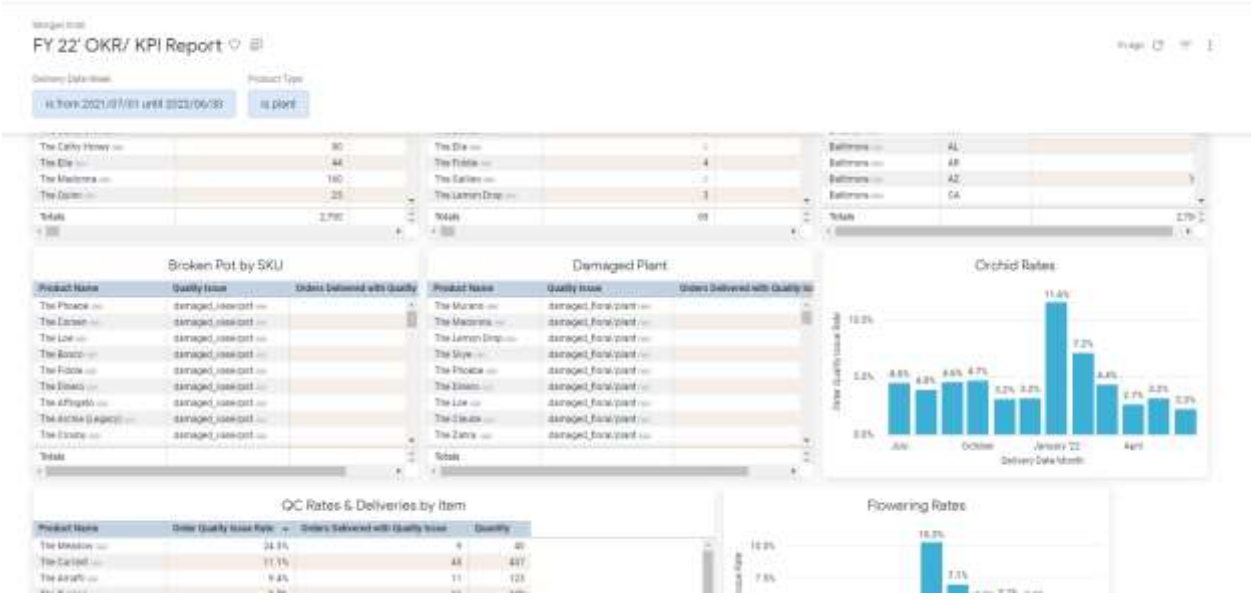
- Change default filter from static begin/end date range to dynamic (i.e. prior 12 complete months).
- Add Dashboard to schedule in order to cache with appropriate datagroup. (LoolML-level cache)

- Reviewed Henry utility installation, output manipulation, and resulting reports.

- Demonstrated the use of Spectacles utility for SQL validation of changes to LookML prior to deployment.

Attachment III: OKR/KPI Investigation

FY 22' OKR/KPI Report - Observations...



1. Report has 33 visualizations, Looker recommended max is 25. Suggest breakup to summary/detail, or product breakouts - i.e. Orchids have several visuals.
2. Static time filter should be replaced with dynamic filter (i.e. last 12 months).

Attachment IV - Miscellaneous Looker Tips

Subject	Item	Description/References
User Communication	Homepage	Here is documentation on how you can set a custom homepage in your Looker instance. The documentation describes how you can leverage the pre-built homepage or you can set the default homepage to a specific board, folder, or a Markdown file (such as a README or document file in a project). Of course you can also set up links that link out to existing tools such as Sharepoint or Confluence if that is where you'd like to host the documentation.
User Communication	Homepage/ Admin side panel	If you decide to leverage the pre-built homepage, you can take advantage of this admin side panel feature where you can push announcements to your users and/or link to training and internal documentation.
User Communication	Custom Help Menu	Another place to plug internal resources is in the the drop-down Help menu at the top right corner of the Looker interface, read how to do so here.
Search Utility	Search Criteria	'Sql_table_name' - surfaces sql table based view names.
Search Utility	Search Criteria	'%' - surfaces 'liquid commands'
Search Utility	Search Criteria	Bind filters: - None found.

Attachment V - Looker/BQML (plus public weather data!!) Example

Github Looker Retail Demo

https://github.com/looker-open-source/block-retail/tree/dev-david-brinegar-dwr5

Summary

- Contains two BigQuery Machine Learning (BQML) models to:
    - create dynamic customer clusters based on their shopping patterns
    - generate stock/sales predictions at the item-store-week level

Additional Insight Section

This project is built against a transaction-item-level table to deliver dashboards and insights that are useful to various teams in a retail organization:

- Regional and store managers
- Merchandising and planning
- CRM and customer teams
- eCommerce teams
- Fraud detection for delivery

Optimized for Google BigQuery, it uses BigQuery nested tables and partition/cluster keys to optimize performance.

Content Details

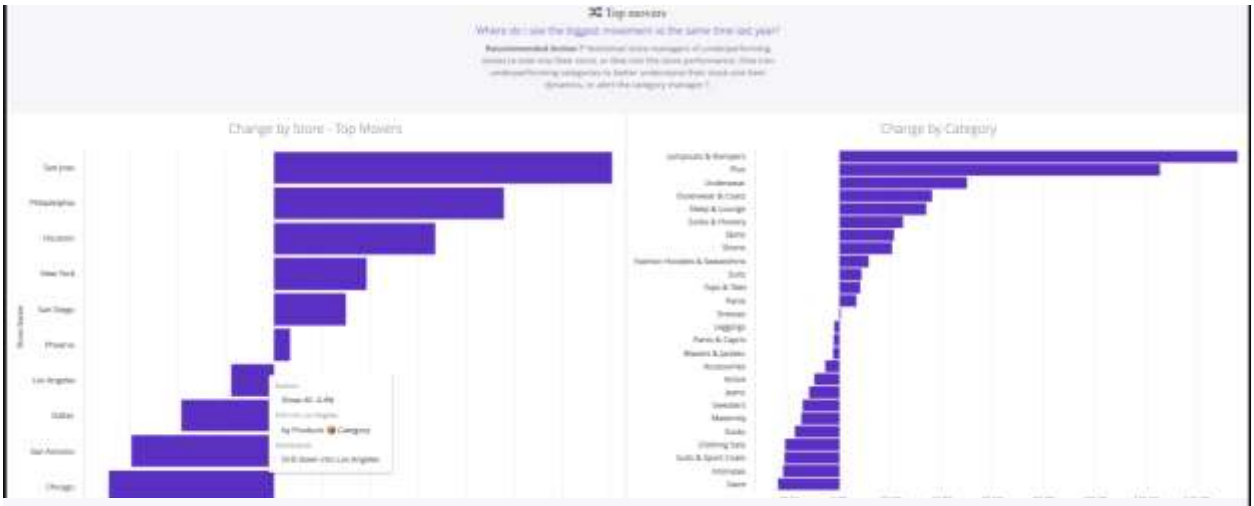
Required Tables:

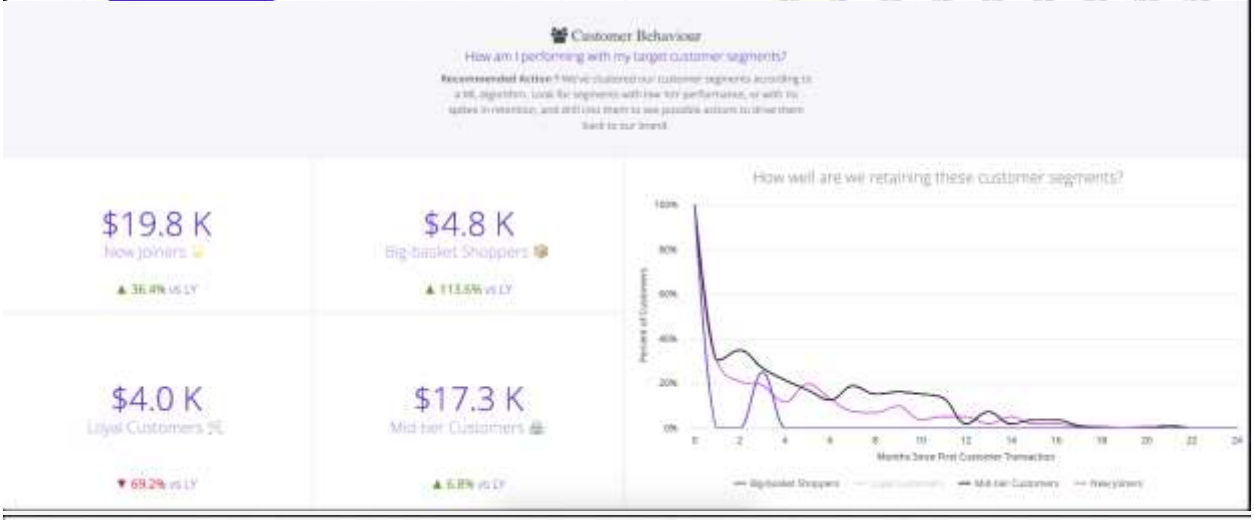
- Transaction-level table (by transaction ID by store by item by customer)
- Store lookup (dim) table
- Item lookup (dim) table

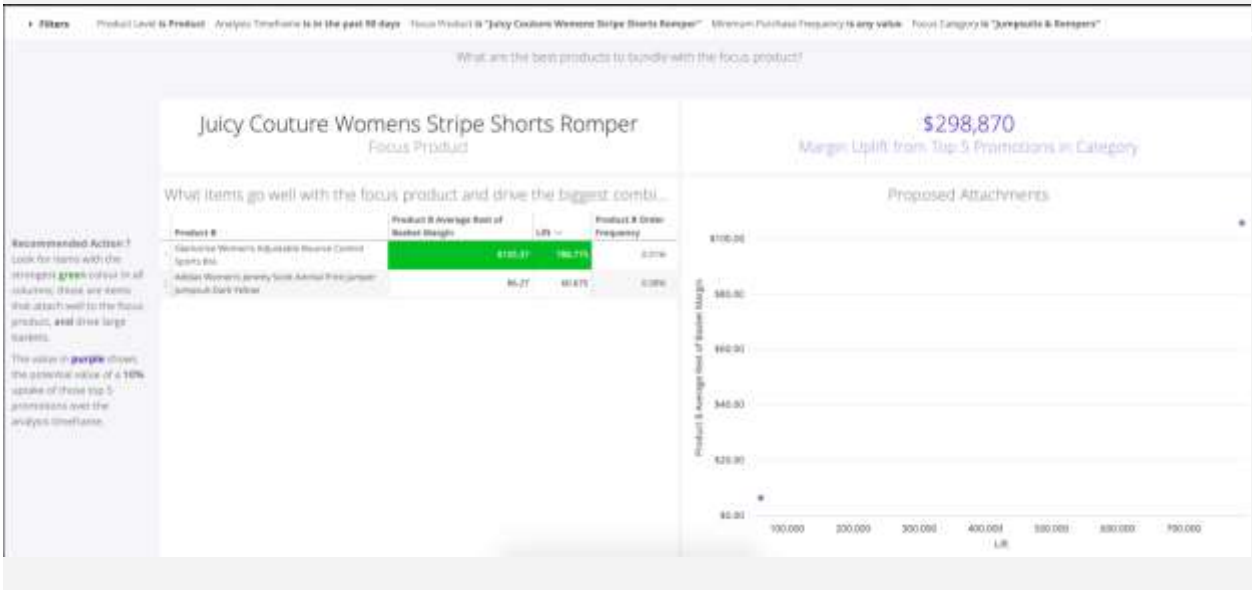
Derives customer info from the transaction table.

Uses BigQuery's public global GHCN weather data.

Screenshots:

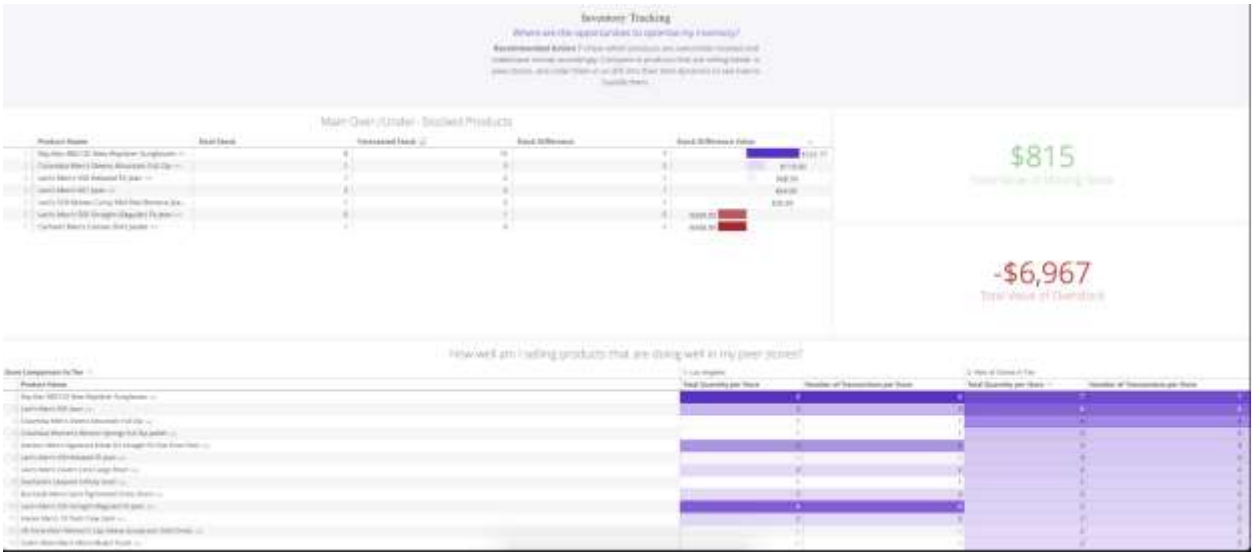

Change by Store - Top Movers
Change by Category


Customer Behaviour
How am I performing with my target customer segments?
$19.8 K
New Joiners
▲ 36.4% vs LY
$4.8 K
Big Basket Shoppers
▲ 113.6% vs LY
$4.0 K
Loyal Customers
▼ 69.2% vs LY
$17.3 K
Mid-tier Customers
▲ 6.8% vs LY
How well are we retaining these customer segments?


What are the best products to bundle with the focus product?
Juicy Couture Womens Stripe Shorts Romper
Focus Product
$298,870
Margin Uplift from Top 5 Promotions in Category
What items go well with the focus product and drive the biggest combi...
Proposed Attachments


Finally, what are my least popular items in that timeframe?
Product Rationalisation
Product A
Product A Order Frequency
Product A Average Basket Margin
Product A Percent Purchased By Loyalty Customer
Product A Percent Customer Exclusivity
Product A Image
Recommended Action ?
Items with a strong red colour are prime candidates for removal from our inventory. They are items that:
are not sold frequently
do not drive large baskets
are not preferred by our key customer segments
will not cause us to lose many non-loyal customers if removed


Peer Store Comparison
How well am I doing vs my peer stores?
YoY Sales
YoY Transaction Count
Weather Trend vs Sales Trend
Weather Trend vs Sales Trend - Peer Stores


Inventory Tracking
Where are the opportunities to optimise my inventory?
Main Over / Under - Stocked Products
$815
-$6,967
How well am I selling products that are doing well in my peer stores?