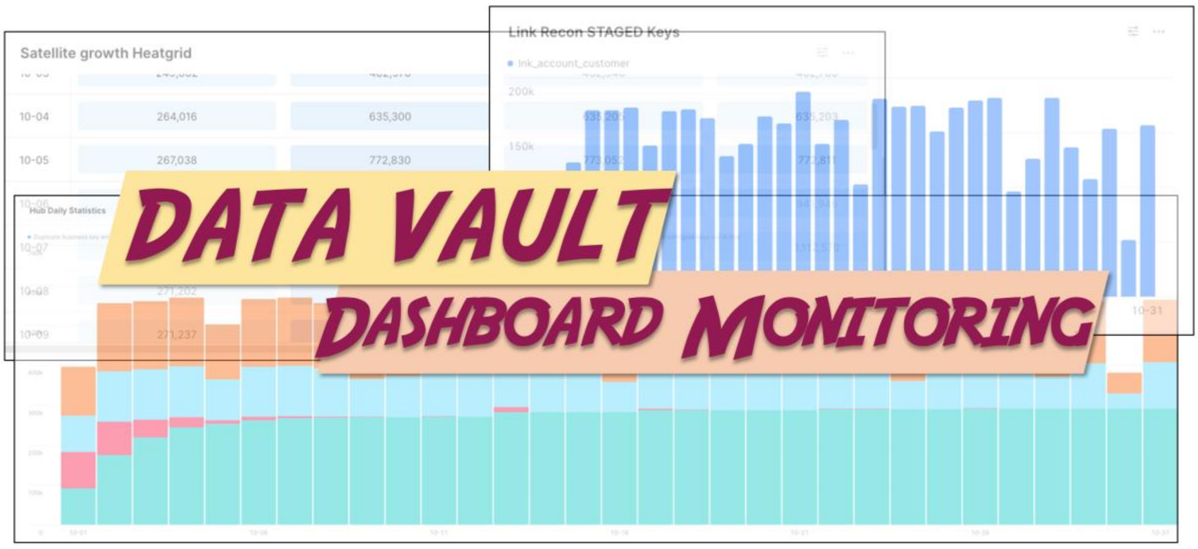


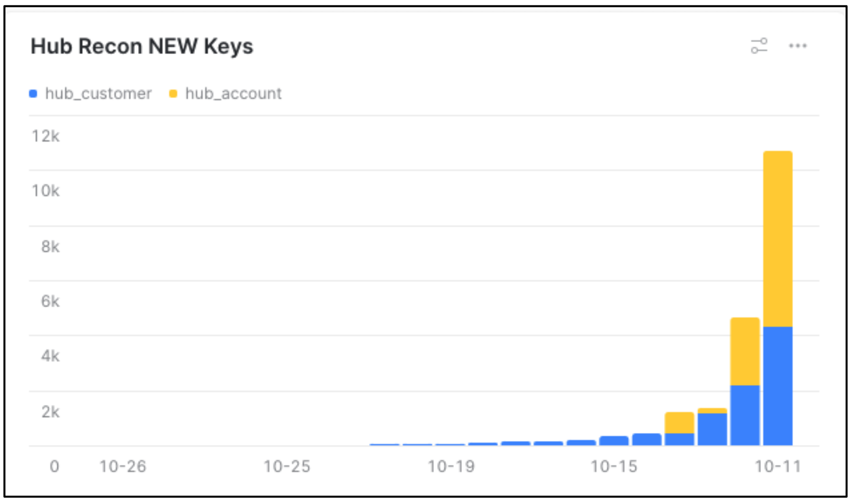
Data Vault Dashboard Monitoring

Following on from Data Vault Automated testing article (see: bit.ly/3dUHPIS) the output of which is captured in reconciliation tables that you can use to monitor the growth of the platform *and* where errors might occur. Automated testing occurs as soon as the data vault artefact or *artefacts* are loaded and should represent as close to real-time what is happening on the platform. Loading the reconciliation metrics in fact follow the same principles for loading a Data Vault 2.0 itself, they are simple, repeatable, and INSERT-ONLY. You can also judge that many of the parameters used for automated testing would be the same parameters used to stage and load data vault artefacts.
This is not a prescriptive post, rather it is what you can do with Snowflake's Snowsight (see: bit.ly/3CteXBf) today to monitor your Data Vault platform. Everything discussed below is custom built and if you have additional reconciliation requirements then these templates serve as a good base to expand upon.

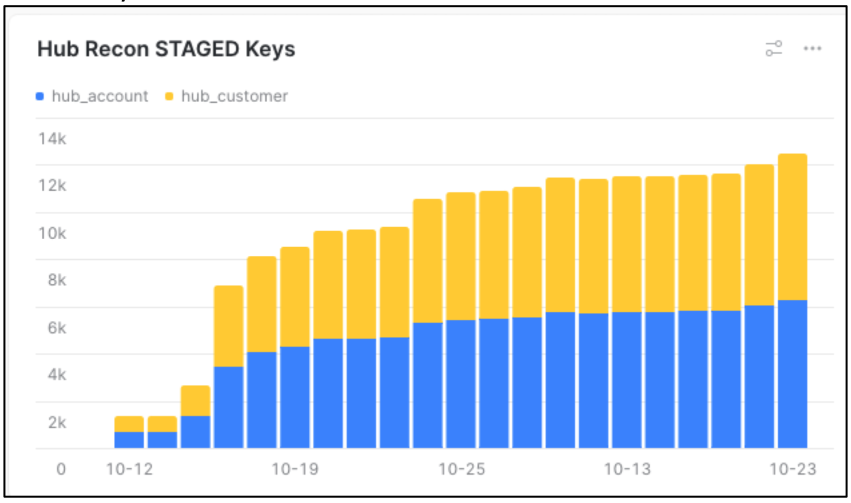
Basic counts

Counting certain attributes of the target Data Vault entities by target type provides some very powerful metrics on what is happening on the platform. Some of these statistics are provided for free given Snowflake's architecture around metadata cache, other statistics can be optimised using Snowflake unique capabilities such as deploying Snowflake Steams on the data vault table themselves to count the number of new records since the last load.

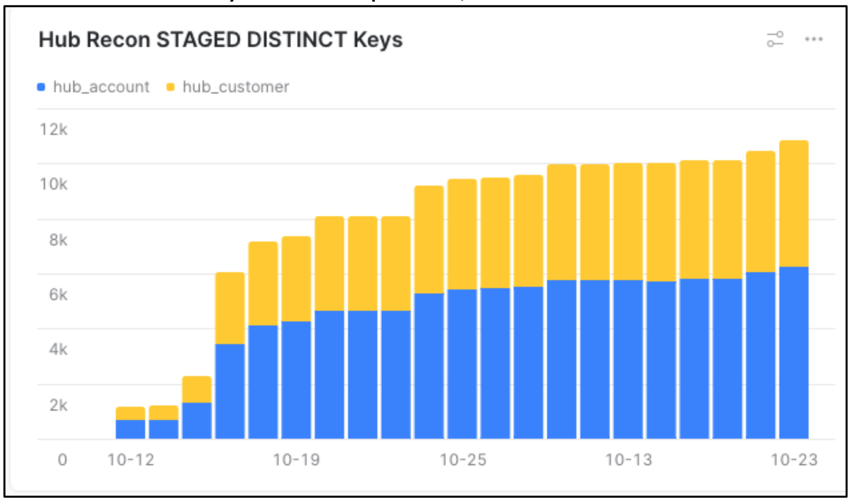
- Hub – counting keys
    - **New** hash key count after load – based on hub recon stream



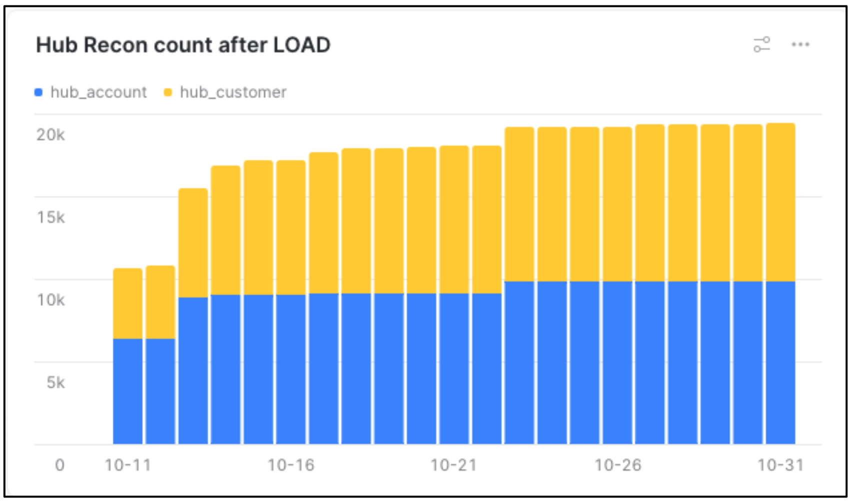
- **Staged** hash key count – uses Snowflake Cache



- **Staged Distinct** hash key count – optional, does not use Snowflake Cache



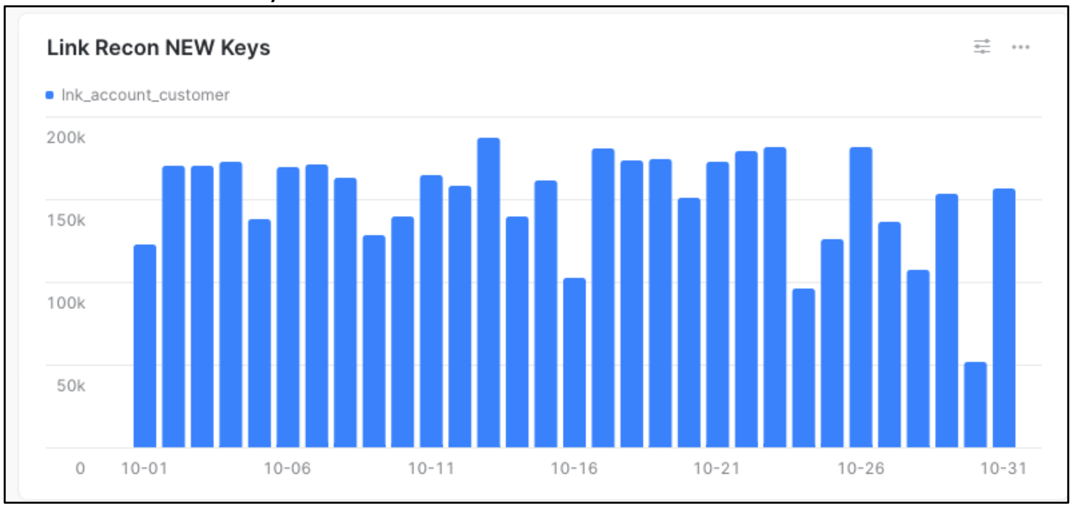
- **Total** hub table count after load - uses Snowflake Cache



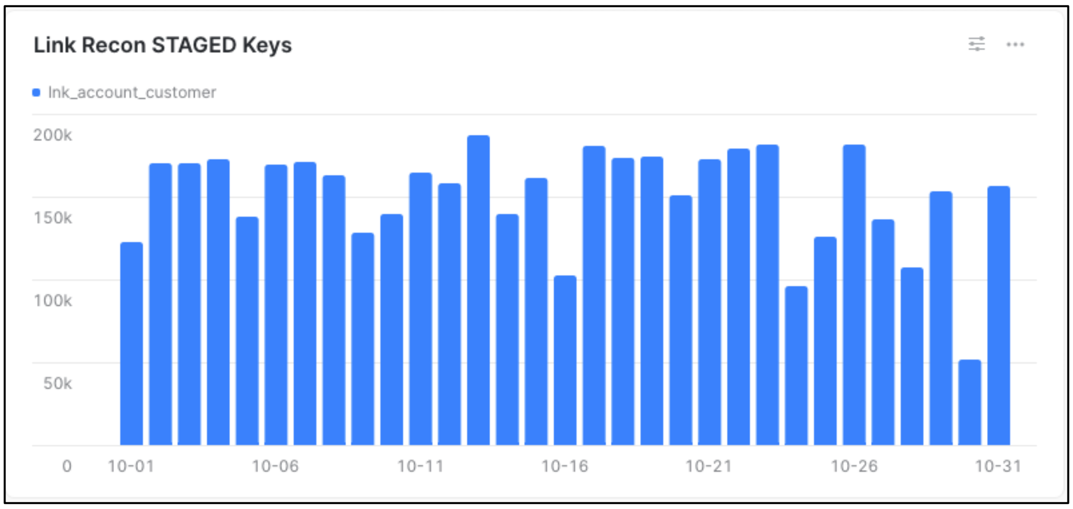
Code with all the stats, use the portion needed.

```
select tablename
, source_tablename
, loaddate
, rundate
, HUB_SKEY_SGTG_NCNT as new_hub_hash_key_count
, HUB_SKEY_SGTG_SCNT as staged_hub_hash_key_count
, HUB_SKEY_SGTG_DCNT as distinct_staged_hub_hash_key_count
, HUB_SKEY_SGTG_TOTAL as hub_table_after_load_count
, HUB_SKEY_SGTG_ERR as missing_hubhash_key_count
from datawarehouse.utilities.reconcile_hub_reconciliation_errors
where loaddate = :daterange
```

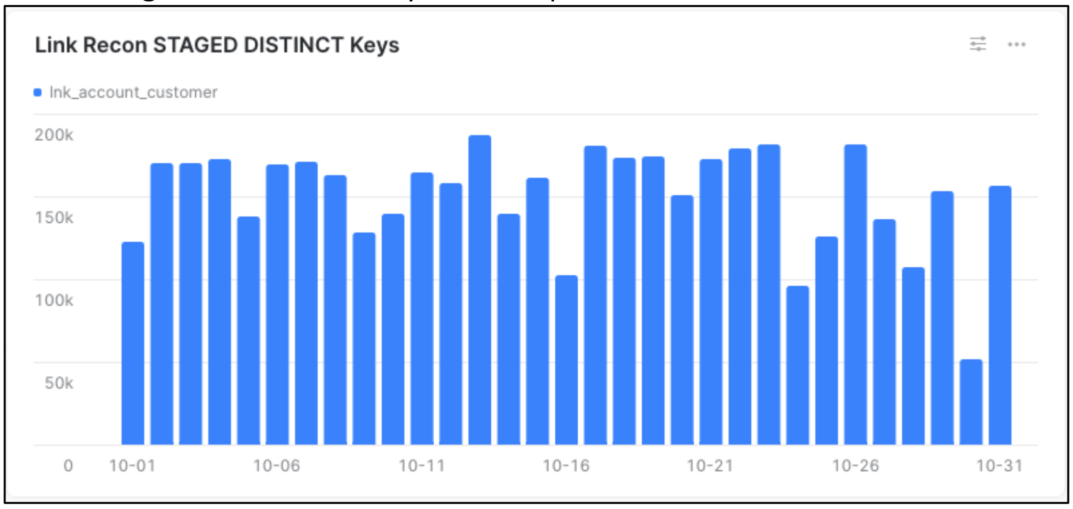
- Link – counting relationships
  - **New** hash key count after load – base this on link recon stream



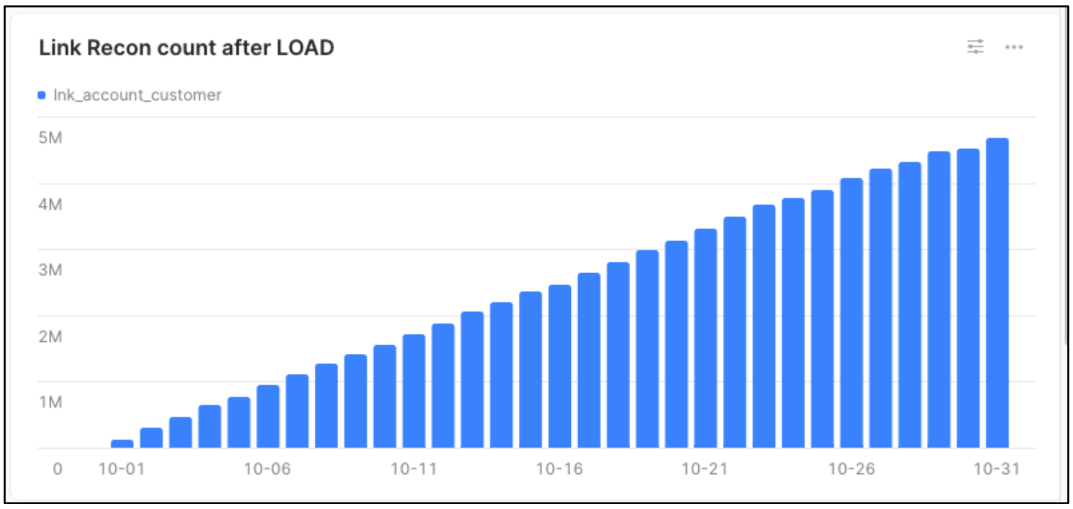
  - **Staged** hash key count – uses Snowflake Cache



- **Staged Distinct** hash key count – optional does not use Snowflake Cache



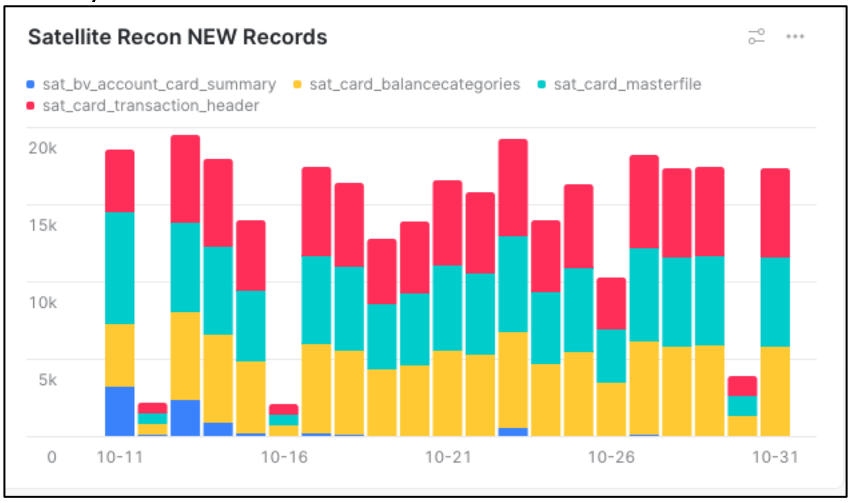
- **Total** link table count after load - uses Snowflake Cache



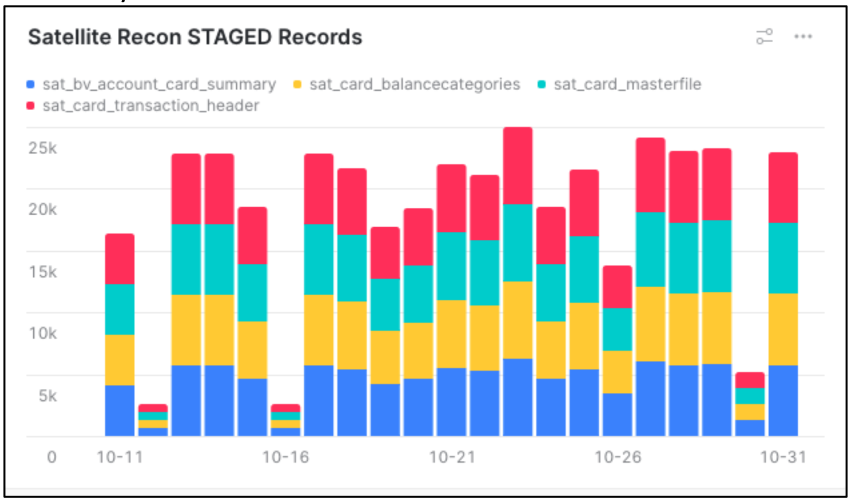
Code with all the stats, use the portion needed.

```
select tablename
, source_tablename
, loaddate
, rundate
, LNK_SKEY_SGTG_NCNT as new_lnk_hash_key_count
, LNK_SKEY_SGTG_SCNT as staged_lnk_hash_key_count
, LNK_SKEY_SGTG_DCNT as distinct_staged_lnk_hash_key_count
, LNK_SKEY_SGTG_TOTAL as lnk_table_after_load_count
, LNK_SKEY_SGTG_ERR as missing_lnkhash_key_count
from datawarehouse.utilities.reconcile_lnk_reconciliation_errors
where loaddate = :daterange
```

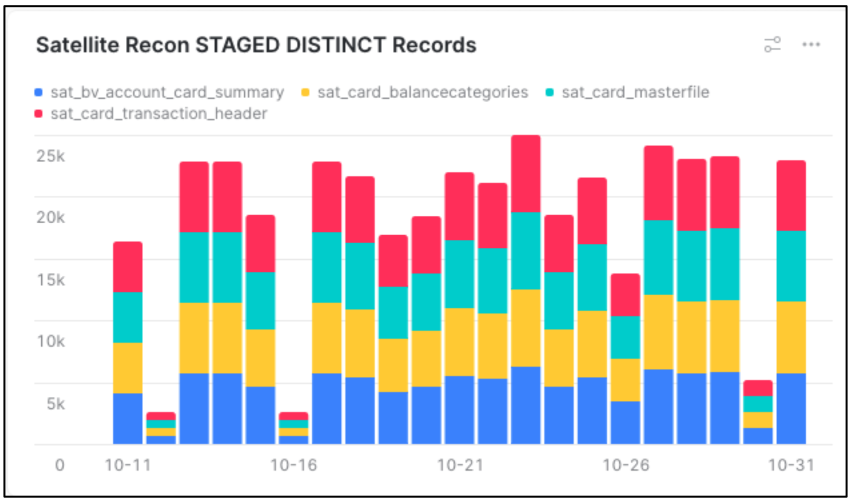
- Satellite – counting descriptions
  - **New** hash key + load date count after load – base this on satellite recon stream



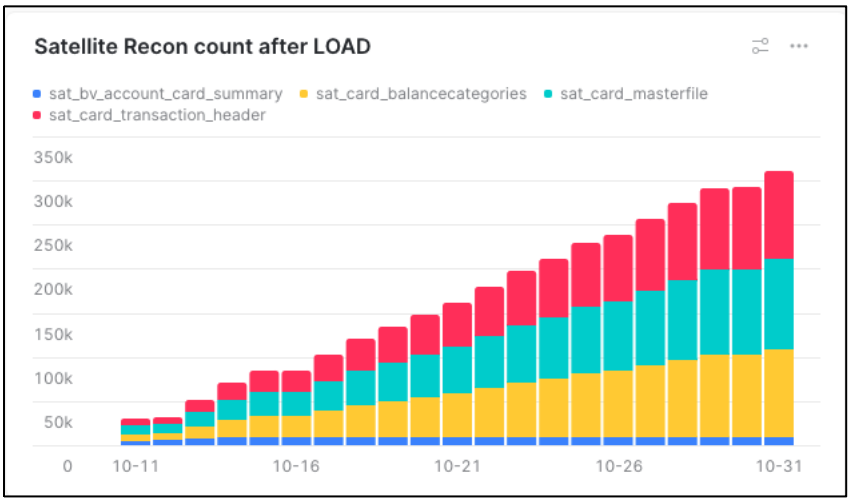
  - **Staged** hash key + load date count – uses Snowflake Cache



  - **Staged Distinct** hash key + load date count – optional, does not use Snowflake Cache



- **Total** satellite table count after load - uses Snowflake Cache



Code with all the stats, use the portion needed.

```
select tablename
, source_tablename
, loaddate
, rundate
, SAT_SKEY_SGTG_NCNT as new_sat_record_count
, SAT_SKEY_SGTG_SCNT as staged_sat_record_count
, SAT_SKEY_SGTG_DCNT as distinct_staged_sat_record_count
, SAT_SKEY_SGTG_TOTAL as sat_table_after_load_count
, SAT_SKEY_SGTG_ERR as missing_sat_records_count
from datawarehouse.utilities.reconcile_sat_reconciliation_errors
where loaddate = :daterange
```

Error checks

Within Snowflake you can define referential integrity constraints between tables, but Snowflake does not enforce them (NOT NULL the exception). Instead, a simple check between related content and ensuring duplicates have not been introduced. Error tracking is split into two parts,

a. Are there errors now?

```
select tablename
, loaddate
, hub_skey_dupe_err
from datawarehouse.utilities.reconcile_hub_duplicate_errors
where loaddate = :daterange
qualify rank() over (partition by loaddate order by rundate desc) = 1
```

b. Were there errors before?

```
select tablename
, loaddate
, hub_skey_dupe_err
from datawarehouse.utilities.reconcile_hub_duplicate_errors
where loaddate = :daterange;
```

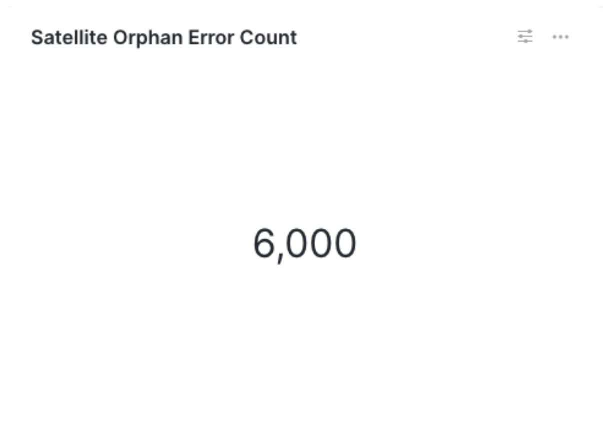
Once errors are corrected the answer to question (a) disappears and question (b) can be used to trace how often they occur

- Duplicate checking
  - Hubs - duplicate surrogate key and business key(s) error count
  - Links – Duplicate link surrogate key and hub key error count
  - Satellites - duplicate surrogate key + load date error count

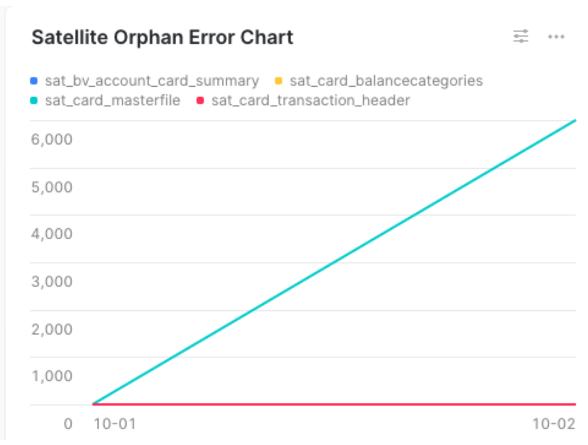
- Staged reconciliation checking
  - Hubs - **missing** surrogate key and business key error count
  - Links – **missing** surrogate key and hub hash keys error count
  - Satellites - **missing** surrogate key and record hash error count based on satellite current view

- Orphan checking
  - Links – **missing** parent keys in hub tables, base this on link orphan streams
  - Satellites - missing surrogate key error count, base this on satellite orphan stream

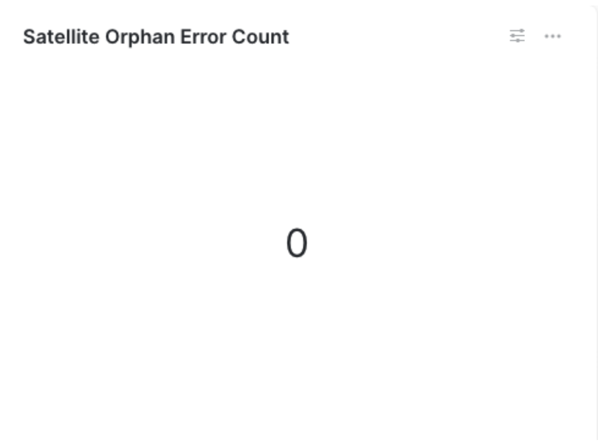
Error detected


Satellite Orphan Error Count

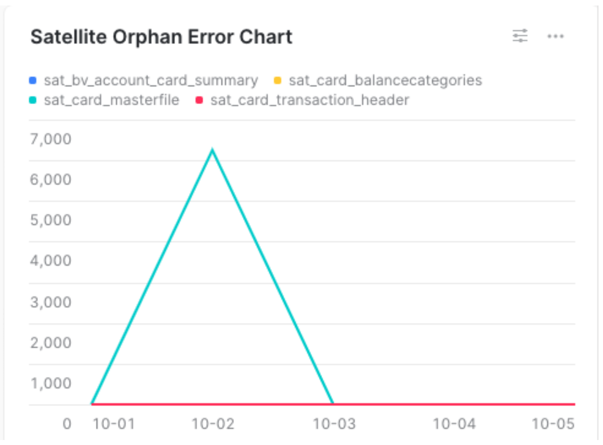
6,000



Error corrected


Satellite Orphan Error Count

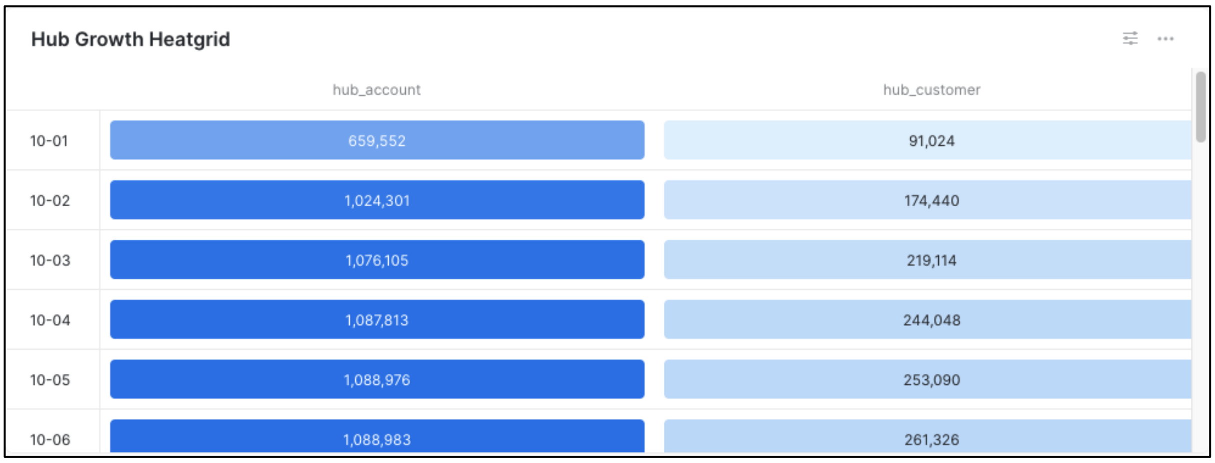
0



Heatgrid Growth

Snowsight has a heat grid that depending on the count will be darker by the higher counts, these samples are based on target table counts.

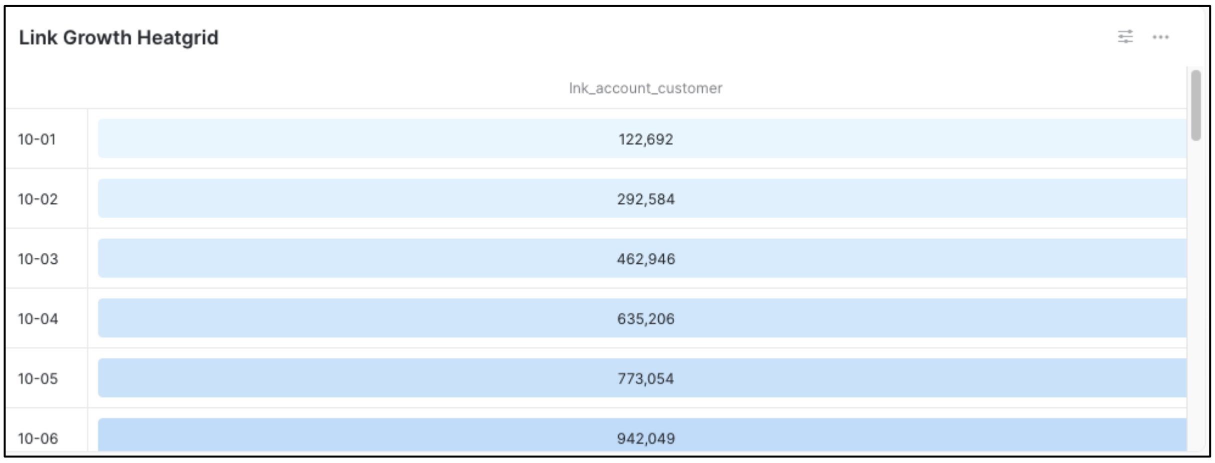
Hubs


Hub Growth Heatgrid

	hub_account	hub_customer
10-01	659,552	91,024
10-02	1,024,301	174,440
10-03	1,076,105	219,114
10-04	1,087,813	244,048
10-05	1,088,976	253,090
10-06	1,088,983	261,326

```
select tablename
, loaddate
, sum(hub_skey_sgtg_total) as hub_skey_sgtg_total
from reconcile_hub_reconciliation_errors
where loaddate =: daterange
group by 1, 2
```

Links

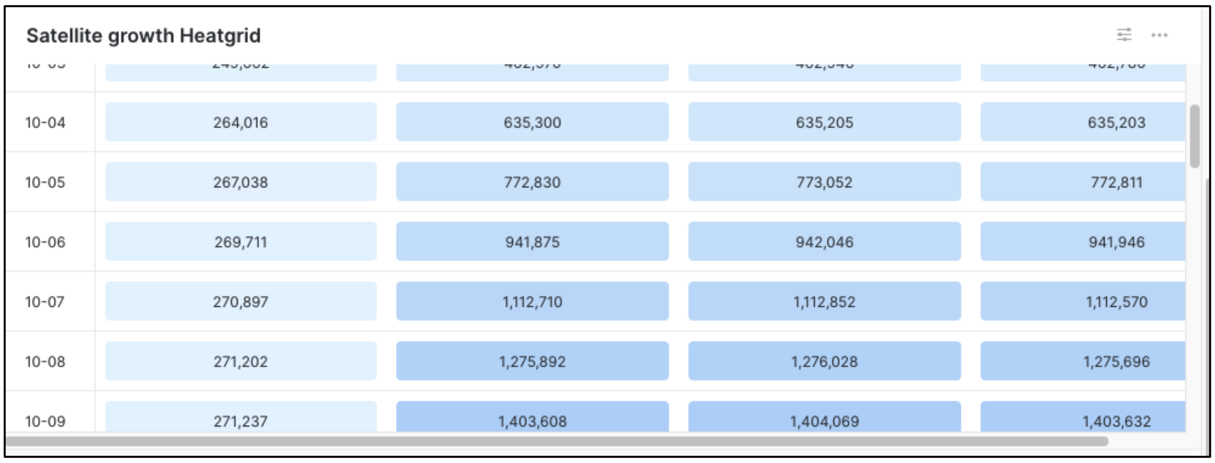


Link Growth Heatgrid

	lnk_account_customer
10-01	122,692
10-02	292,584
10-03	462,946
10-04	635,206
10-05	773,054
10-06	942,049

```
select tablename
, loaddate
, sum(lnk_skey_sgtg_total) as lnk_skey_sgtg_total
from reconcile_lnk_reconciliation_errors
where loaddate = :daterange
group by 1, 2
```

Satellites



Satellite growth Heatgrid

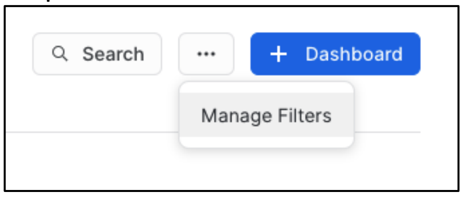
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10-04	264,016	635,300	635,205	635,203
10-05	267,038	772,830	773,052	772,811
10-06	269,711	941,875	942,046	941,946
10-07	270,897	1,112,710	1,112,852	1,112,570
10-08	271,202	1,275,892	1,276,028	1,275,696
10-09	271,237	1,403,608	1,404,069	1,403,632

```
select tablename
, loaddate
, sum(sat_skey_sgtg_total) as sat_skey_sgtg_total
from reconcile_sat_reconciliation_errors
where loaddate = :daterange
group by 1, 2
```

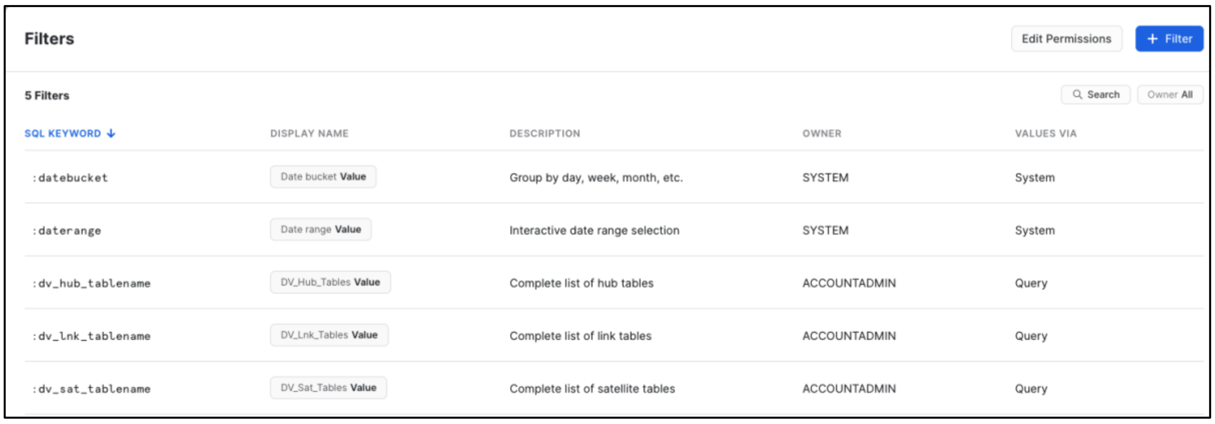
Per artefact analysis

Snowsight has the ability to administer your own custom filters over and above the filters already provided, :daterange and :datebucket. Snowsight will detect the presence of valid filters in your dashboard and add the appropriate dropdown filters on the top left of the dashboard. To create your own filters and use them in your dashboards start by managing filters,

Step 1



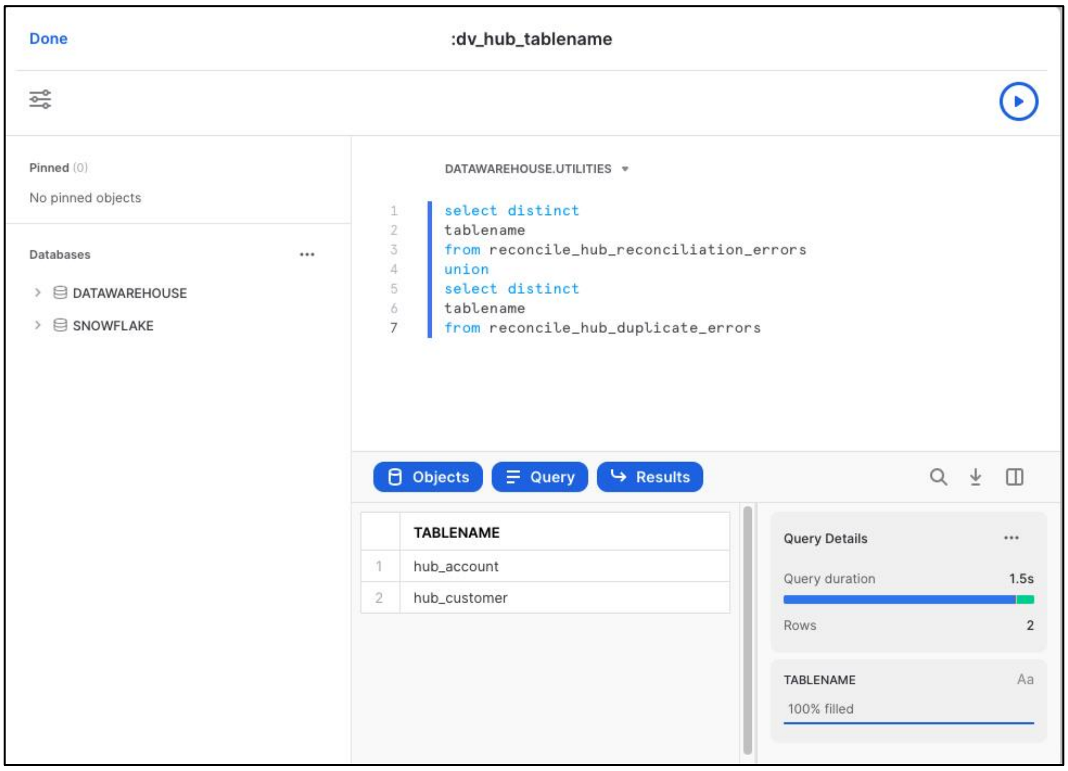
Edit or add a new filter



5 Filters

SQL KEYWORD ↓	DISPLAY NAME	DESCRIPTION	OWNER	VALUES VIA
:datebucket	Date bucket Value	Group by day, week, month, etc.	SYSTEM	System
:daterange	Date range Value	Interactive date range selection	SYSTEM	System
:dv_hub_tablename	DV_Hub_Tables Value	Complete list of hub tables	ACCOUNTADMIN	Query
:dv_lnk_tablename	DV_Lnk_Tables Value	Complete list of link tables	ACCOUNTADMIN	Query
:dv_sat_tablename	DV_Sat_Tables Value	Complete list of satellite tables	ACCOUNTADMIN	Query

Step 2 – Decide the type of filter, dynamic or static, we're going dynamic

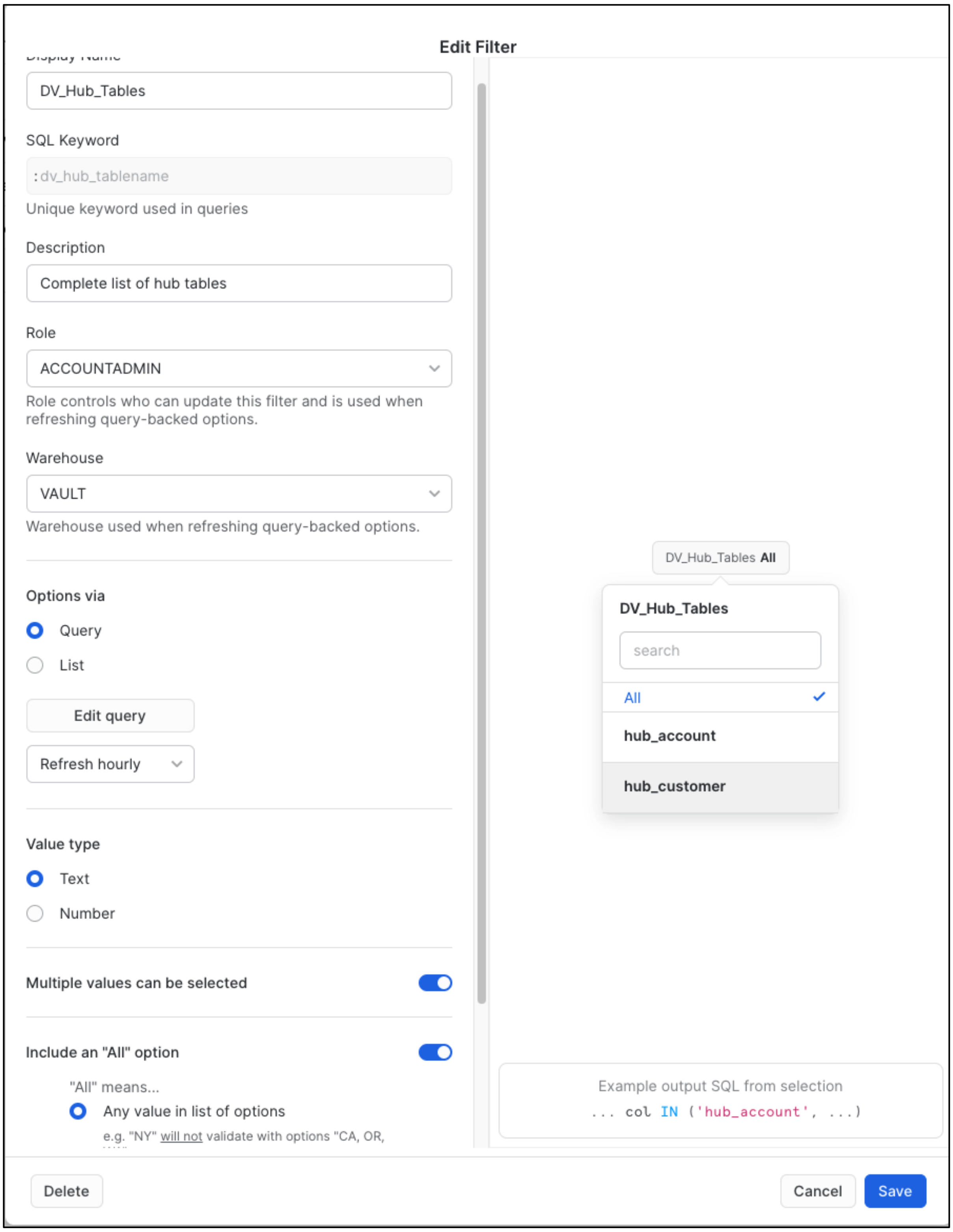


```sql
select distinct
tablename
from reconcile_hub_reconciliation_errors
union
select distinct
tablename
from reconcile_hub_duplicate_errors
```

	TABLENAME
1	hub_account
2	hub_customer

Query Details
Query duration 1.5s
Rows 2
TABLENAME Aa
100% filled

Step 3 – Define other Filter criteria like being able to select multiple options



Edit Filter

SQL Keyword
:dv_hub_tablename
Unique keyword used in queries

Display Name: DV_Hub_Tables

Description: Complete list of hub tables

Role: ACCOUNTADMIN
Role controls who can update this filter and is used when refreshing query-backed options.

Warehouse: VAULT
Warehouse used when refreshing query-backed options.

Options via
[✓ Query]
List

Edit query
Refresh hourly

Value type
[✓ Text]
Number

Multiple values can be selected [✓]

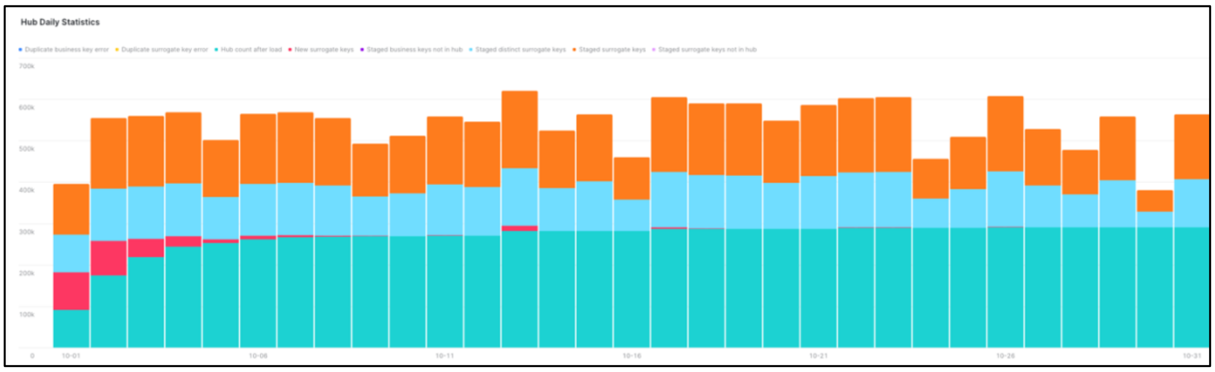
Include an "All" option [✓]
"All" means...
[✓ Any value in list of options]
e.g. "NY" will not validate with options "CA, OR,

DV_Hub_Tables All

DV_Hub_Tables
search
All ✓
hub_account
hub_customer

Example output SQL from selection
```
... col IN ('hub_account', ...)
```

Use the filter to do artefact by artefact analysis



Code to unpivot and use for filtering, hubs

```
select tablename
, loaddate
, stats
, case when counts = 'HUB_SKEY_SGTG_DCNT' then 'Staged distinct surrogate keys'
    when counts = 'HUB_SKEY_SGTG_SCNT' then 'Staged surrogate keys'
    when counts = 'HUB_SKEY_SGTG_NCNT' then 'New surrogate keys'
    when counts = 'HUB_SKEY_SGTG_TOTAL' then 'Hub count after load'
    when counts = 'HUB_SKEY_SGTG_ERR' then 'Staged surrogate keys not in hub'
    when counts = 'HUB_BKEY_SGTG_ERR' then 'Staged business keys not in hub'
    else null
    end as counts
from reconcile_hub_reconciliation_errors
unpivot(stats for counts in (hub_skey_sgtg_ncnt, hub_skey_sgtg_scnt, hub_skey_sgtg_dcnt,
hub_skey_sgtg_total, hub_skey_sgtg_err, hub_bkey_sgtg_err))
where loaddate = :daterange
and tablename = :dv_hub_tablename
union all
select tablename
, loaddate
, stats
, case when counts = 'HUB_SKEY_DUPE_ERR' then 'Duplicate surrogate key error'
    when counts = 'HUB_BKEY_DUPE_ERR' then 'Duplicate business key error'
    else null
    end as counts
from reconcile_hub_duplicate_errors
unpivot(stats for counts in (hub_skey_dupe_err, hub_bkey_dupe_err))
where loaddate = :daterange
and tablename = :dv_hub_tablename
order by loaddate, tablename;
```

Unpivot Links

```
select tablename
, loaddate
, stats
, case when counts = 'LNK_SKEY_SGTG_DCNT' then 'Staged distinct link keys'
    when counts = 'LNK_SKEY_SGTG_SCNT' then 'Staged link keys'
    when counts = 'LNK_SKEY_SGTG_NCNT' then 'New link keys'
    when counts = 'LNK_SKEY_SGTG_TOTAL' then 'Link count after load'
    when counts = 'LNK_SKEY_SGTG_ERR' then 'Staged link keys not in link'
    when counts = 'LNK_HKEY_SGTG_ERR' then 'Staged hub keys not in link'
    else null
    end as counts
from reconcile_lnk_reconciliation_errors
unpivot(stats for counts in (lnk_skey_sgtg_ncnt, lnk_skey_sgtg_scnt, lnk_skey_sgtg_dcnt,
lnk_skey_sgtg_total, lnk_skey_sgtg_err, lnk_hkey_sgtg_err))
where loaddate = :daterange
and tablename = :dv_lnk_tablename
union all
select tablename
, loaddate
, stats
, case when counts = 'LNK_SKEY_DUPE_ERR' then 'Duplicate link key error'
    when counts = 'LNK_HKEY_DUPE_ERR' then 'Duplicate hub key error'
    else null
    end as counts
from reconcile_lnk_duplicate_errors
unpivot(stats for counts in (lnk_skey_dupe_err, lnk_hkey_dupe_err))
where loaddate = :daterange
and tablename = :dv_lnk_tablename
union all
select tablename
, loaddate
, stats
, case when counts = 'LNK_SKEY_ORPH_ERR' then 'Orphan link key error'
    else null
    end as counts
from reconcile_lnk_referential_errors
unpivot(stats for counts in (lnk_skey_orph_err))
where loaddate = :daterange
and tablename = :dv_lnk_tablename
order by loaddate, tablename;
```

And Satellites

```
select tablename
, loaddate
, stats
, case when counts = 'SAT_SKEY_SGTG_DCNT' then 'Staged distinct satellite records'
    when counts = 'SAT_SKEY_SGTG_SCNT' then 'Staged satellite records'
    when counts = 'SAT_SKEY_SGTG_NCNT' then 'New satellite records'
    when counts = 'SAT_SKEY_SGTG_TOTAL' then 'Satellite count after load'
    when counts = 'SAT_SKEY_SGTG_ERR' then 'Staged surrogate keys not in satellite'
    when counts = 'SAT_HDIF_SGTG_ERR' then 'Staged record hash and key not in satellite'
    else null
    end as counts
from reconcile_sat_reconciliation_errors
unpivot(stats for counts in (sat_skey_sgtg_ncnt, sat_skey_sgtg_scnt, sat_skey_sgtg_dcnt,
sat_skey_sgtg_total, sat_skey_sgtg_err, sat_hdif_sgtg_err))
where loaddate = :daterange
and tablename = :dv_sat_tablename
union all
select tablename
, loaddate
, stats
, case when counts = 'SAT_SKEY_DUPE_ERR' then 'Duplicate surrogate key and load date error'
    else null
    end as counts
from reconcile_sat_duplicate_errors
unpivot(stats for counts in (sat_skey_dupe_err))
where loaddate = :daterange
and tablename = :dv_sat_tablename
union all
select tablename
, loaddate
, stats
, case when counts = 'SAT_SKEY_ORPH_ERR' then 'Orphan satellite key error'
    else null
    end as counts
from reconcile_sat_referential_errors
unpivot(stats for counts in (sat_skey_orph_err))
where loaddate = :daterange
and tablename = :dv_sat_tablename
order by loaddate, tablename;
```

The views expressed in this article are that of my own, you should test implementation performance before committing to this implementation. The author provides no guarantees in this regard.