

10 MARCH 2021

### How to Add a Tenant View, OLAP View, or OLAPTS Table for Extract

5.21.11 - CreditLens™ Analytics and Reporting Configuration Guide

### **Contact**

Moody's Analytics Support

MA\_TechSupport@moodys.com

+1-212-553-1653

www.moodysanalytics.com/support

### **Documentation**

Information Web: https://information.moodysanalytics.com/

Customer Portal: https://moodys-analytics.force.com/customers/sfc/#search?tn=Content

| CONFIDENTIAL  |
|---|
| This document contains information proprietary to Moody's Analytics, Inc. ("Moody's") and its affiliates. This information may not be reproduced, disclosed, or used in whole or in part without the express written permission of Moody's.   |
| TRADEMARKS  |
| Moody's Analytics, Moody's, and all other names, logos, and icons identifying Moody's Analytics and/or its products and services are trademarks of Moody's Analytics, Inc. or its affiliates. Third-party trademarks referenced herein are the property of their respective owners. |
|   |
|   |
|   |

| Table of Contents |  |  |
|-------------------|--|--|
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |
|                   |  |  |

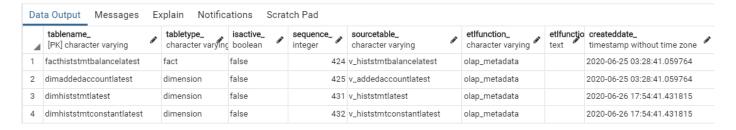
To read a tenant view or join that view from the tenant directly to an olap table, such as v\_histstmtbalancelatest, you can add the table directly into olap without causing any ETL transformation to take place. These tables can be added and flagged as system tables and set to **isactive\_=false**.

You can run this query to see what tables are inactive in the olap\_etl\_control table and are also flagged as system tabletypes.

### Query:

# codeblock select \* from olapts.olap\_etl\_control where sourcetable\_ ilike 'v\\_%' and etlfunction\_= 'olap\_ metadata'; -- as of 5.21.11 you can run the following view below, which gives the same information as the previous query - select \* from v olapts latestviews;

### Results:



Complete the following procedures to create a new view inside of tenantdb called v\_histstmtlatest, and expose it tothe olap database for querying and data extract.

Query: Create a view in tenantdb.

```
codeblock
-- tenant prefix is not needed but prevents you from creating the view in olap accidentally --
DROP VIEW IF EXISTS tenant.v histstmtlatest;
analyze tenant.historicalstatement;
create or replace view tenant.v_histstmtlatest as
select
    h.pkid as v histstmtlatestid -- create pkid and rename it as viewname||id
      --- alternatively you can use h.id as primarykey and rename it same as viewname||id ---
     --- alternatively you can use h:id |\cdot|h.versionid as primarykey or use any set of columns to set
granularity of table such as pkid ||accountid and name as viewname||id
    ,h.pkid :: varchar as pkid -- pull pkid for transformation if this is latestversion view
    ,h.id :: varchar as historicalstatementid -- pull the id from tenant table and name it as
tenant.tablename||id
-- add several jsonb columns here and alphabetize them in lowercase order
    ,(h.jsondoc ->> 'AccountId' ):: numeric as accountid
    ,(h.jsondoc ->> 'ColumnA' )::text as columna
   ,(h.jsondoc_ ->> 'ColumnZ' )::text as columnz
 -- add all base table context attributes you want for your view here
```

```
,h.wfid :: varchar
    ,h.taskid :: varchar
    ,h.versionid ::int4
    ,h.isdeleted ::boolean
    ,h.islatestversion ::boolean
    ,h.baseversionid ::int4
    ,h.contextuserid :: varchar
    ,h.isvisible ::boolean
    ,h.isvalid ::boolean
    ,h.snapshotid ::int4
    ,h.t :: varchar
    ,h.createdby :: varchar
    ,h.createddate :: timestamp
    ,h.updatedby :: varchar
    ,h.updateddate :: timestamp
-- add these 2 columns to be used for incremental transformation and incremental dataextracts --
   ,( case when h.updateddate_> h.createddate_ then h.updatedby_ else h.createdby_ end ):: varc
    as sourcepopulatedby
, GREATEST( h.createddate_,h.updateddate_):: Timestamp as sourcepopulateddate
-- add your FROM clause here - if using isvisible , isvalid , islatestversion and not isdeleted you can
also add the word ONLY to the FROM clause --
    from only historicalstatement h
      where ( h.isvalid and h.isvisible and h.islatestversion )
     ; -- note: and not h.isdeleted this was removed in 20.31.1
```

### Important Notes on View Design:

- The view must be named as v\_someview. This tells the olap code that it is a view that could potentially be converted as a table using the same format.
  - For specific customer views, name it as v\_{customerinitials}viewname such as v\_abctenantfinancialview. This prevents any views Moody's creates from overriding existing customer custom views
- The view must have the first column as v\_someview+id\_, such as v\_someviewid\_, which holds the granularity of the table. It is the primary key and cannot have duplicates.
- The view must have pkid\_ separately, as pkid\_ translates back to the latestversion of data.
- The view must have a column called id\_ that is renamed as sourcetablename+id\_. In this case, it is the historical statement id\_. This allows joining back to the tenant source table using this id\_ value.

### Results:



Query: Select from the view to ensure that it gives the data you want.

### codeblock -- tenant prefix is not needed, but prevents you from creating the view in olap accidentally -select \* from tenant.v\_histstmtlatest limit 5

### Results:

| 4 | v_histstmtlatestid_<br>character varying | pkid_<br>character varying | historicalstatementid_<br>character varying | accountid<br>numeric | colum<br>text | nna 🔒 | columnz<br>text | wfid_<br>character varying | taskid_<br>character varying | versionid_<br>integer | isdeleted_<br>boolean | islatestversion,<br>boolean |
|---|--|----------------------------|---|----------------------|---------------|-------|-----------------|----------------------------|------------------------------|-----------------------|-----------------------|-----------------------------|
| 1 | 83                                       | 83                         | 83 2  | [n                   | ull] [null]   |       | [null]          | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 2                     | false                 | true                        |
| 2 | 90                                       | 90                         | 90 5  | [n                   | ull] [null]   |       | [null]          | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 5                     | false                 | true                        |
| 3 | 89                                       | 89                         | 89 10                                       | [n                   | ull] [null]   |       | [null]          | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 10                    | false                 | true                        |
| 4 | 1  | 1                          | 1 3   | [n                   | ull] [null]   |       | [null]          | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 3                     | false                 | true                        |
| 5 | 2  | 2                          | 2 3   | [n                   | ull] [null]   |       | [null]          | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 3                     | false                 | true                        |

### Switch user/database and log in as olap user in olap database

**Query:** This query screate a view reference in the FDW for olap to query and also create a new table called olapts.facthiststmtlatest.

```
codeblock
select olapts.populate_olapmetadata_views('v_histstmtlatest', 'dim');
```

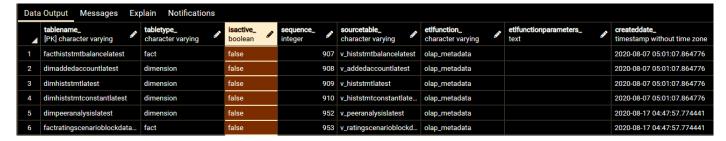
**Results:** The view is now registered as the same view in the olap database and is also registered as an olapts table if needed. This table will be set to inactive by default so as not to slow down the current Time Series job.

### Query:

## codeblock select \* from olapts.olap\_etl\_control where sourcetable\_ ilike 'v\\_%' and etlfunction\_= 'olap\_ metadata'; -- as of 5.21.11, you can run this view below, which gives same information as the previous query - select \* from v\_olapts\_latestviews;

Results: Views are available to query in olap but will not generate an olapts table unless they are set to isactive\_=true.

Note If isactive\_=true, the view will materialize into an olapts table using existing DDL at that time. If the tenant.view is updated later, then this process needs to repeat to generate an updated table format in olapts.



Following are reasons to materialize a view into an olapts table:

- The view has too many rows or takes too long to run in real-time.
- You are unable to generate the data extract file directly from the view.
- The trade off is that the time that it takes executing queries against the view prior to being materialized will be the same time it takes to materialize a view during the next Time Series Job run.
- You are creating an olapts view called "view1" and it joins this view above (*for example*, v\_histstmtlatest to olapts.dimfinancial, olapts.dimentity. joining the results of this view to another olapts tables.
  - Joining an olapts view to olapts tables causes the newly created view ("view1" in previous bullet) to drop from existence or disappear from the olapts database on each run of the TimeSeries ETL process.

Note the following differences in data extracts:

- Extracts of a view are in real-time, as they come directly from the olap link into the tenantdb called Foreign Data Wrapper (FDW).
- Extracts of a materialized olapts table are a snapshot of data from the last run of the Time Series Job.

To materialize a view into an olapts table complete the following query:

Query: Update the row in olapts.olap\_etl\_control where sourcetable\_='v\_yournewviewname'.

### codeblock

```
update olapts.olap etl control set isactive = true where sourcetable = 'v_histstmtlatest' and
etlfunction = 'olap_metadata';

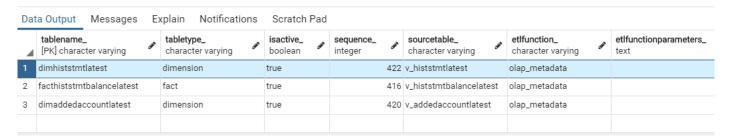
-- as of 5.21.11, you can run the following view, which gives the same information as the previous query --
update v_olapts_latestviews set isactive = true where sourcetable = 'v_histstmtlatest';

-- or to enable all views at once, such as all IPRE views, you can use something like this statement

update olapts.olap etl control set isactive = true where sourcetable ilike 'v\_ipre%' and
etlfunction = 'olap metadata';

-- as of 5.21.11, you can run the following view, which gives same information as the previous query --
update v olapts latestviews set isactive = true;
```

Results: This view materializes into an olapts table on the next Time Series job run.



The best practice is to enable one view at a time, then execute the TimeSeries Job, then check for errors. If the first view is correct, then enable the next one until all are enabled.

Execute the Reporting Database Time Series Job, and check for errors.

If you find any errors, make note of what the error message is, then go in and set the specific view/table back to **inactive** so that the job does not keep erroring out until you can fix your view.

### codeblock

```
update olapts.olap etl control set isactive_= false where sourcetable_ = 'v_histstmtlatest' and
etlfunction = 'olap metadata';
-- as of 5.21.11, you can run the following view, which gives the same information as the previous query --
update v_olapts_latestviews set isactive_= false;
```

If you are creating a brand new tenant view to use in this process, you will need to analyze the view design and set it up so that there are no duplicates in the PrimaryKey, and the granularity of the view is such that it allows for only one unique row for each PrimaryKey regardless of version.

The Primary Key will normally be the id\_ of the underlying table; however, if you have multiple tables involved, you must add key values from each table into a combined PrimaryKey for this view.

For example, if the view joins Entity, HistoricalStatement, Financial tables, then the PrimaryKey would be like this: (entity.pkid\_||':'||historicalstatement.pkid\_||':'||financial.pkid\_) as v\_viewnamepkid.

When using pkid\_ as the primary key, you also need to have the WHERE entity.islatestversion\_ and historical statement.islatestversion\_ and financial.islatestversion\_ filters - to remove duplicate versions from a materialized table that joins entity, historical statement, financial tables together.

If you receive a Success message, then check the new table by selecting from it. In this example, only 20 rows were selected.

### Query:

### codeblock

select \* from olapts.dimhiststmtlatest limit 20; -- this is your view called v\_histstmtlatest converted
into an olapts table.

Results: table output

| Data O | utput Messages Expla                        | in Notifications So        | cratch Pad                                  |                      |                            |                              |                       |
|--------|---|----------------------------|---|----------------------|----------------------------|------------------------------|-----------------------|
|        | dimhiststmtlatestid_ [PK] character varying | pkid_<br>character varying | historicalstatementid_<br>character varying | accountid<br>numeric | wfid_<br>character varying | taskid_<br>character varying | versionid_<br>integer |
| 1      | 1   | 1                          | 1 1   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 2      | 2   | 2                          | 2 1   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 3      | 6   | 6                          | 6 1   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 4      | 7   | 7                          | 7 1   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 5      | 43  | 43                         | 43 1  | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 6      | 4   | 4                          | 4 2   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 2                     |
| 7      | 5   | 5                          | 5 2   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 2                     |
| 8      | 13  | 13                         | 13 1  | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 9      | 8   | 8                          | 8 1   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 10     | 9   | 9                          | 9 1   | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 11     | 10  | 10                         | 10 1  | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 12     | 11  | 11                         | 11 1  | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |
| 13     | 136   | 136                        | 136 1                                       | [null]               | 00000000-0000-0000-0       | 00000000-0000-0000-0         | 1                     |

Now, each time you run the Time Series job, the table will run an UPSERT and either Insert or Update any incremental rows from the view that have changed in tenant, and push them into your olapts materialized table.

This process allows for faster subsequent runs of the TimeSeries job using incremental updates to this new table.

Want to find out more? You can find the most up-to-date product documentation on Information Web (https://information.moodysanalytics.com).