Technical Methods to Ease Migration from Hyperion Interactive Reports

December 11, 2018, ADAL Sprint 187

Office of Systems
Advanced Data Analytics Lab

Patricia Stanton

Table of Contents

Federal agencies can make use of relational database technology, scripting languages such as Python, and restful web services to ease the burden of report migration from one enterprise solution to another.

Overview

Background

Besides the programmatic applications that power the core mission, enterprise-reporting applications are important to provide management information that federal agencies need to make informed decisions when planning goals, workloads, and resources. When one thinks of enterprise report offerings, the vendors that most commonly come to mind are Crystal Reports, Tableau, WebFocus, and Hyperion. The enterprise report landscape has changed key players and solutions quite a bit. Agencies face challenges with migrating from one enterprise report solution to another. There are different reasons an agency must migrate to different solution and it can be because a vendor has decided to discontinue support or the agency has decided that a different product is more in line with its needs.

The Social Security Administration (SSA), over the past decade used Hyperion's Enterprise Performance Management (EPM) System for some its Management Information and Reporting needs. The EPM product had an interesting lifecycle over the last twenty years. It originated from a company called Brio Software who called the product Brioquery. Hyperion purchased Brioquery and rebranded the product as such. Oracle then purchased Hyperion in 2007 and supported the product for over a decade. Oracle has now decided to sunset the product to focus on its other business intelligence solutions.

SSA has over fifty thousand reports housed in EPM. Employees have labeled over three thousand reports as critical reports that must be migrated. With the evolving technical landscape concerning big data and cloud computing, SSA is moving to a mixture of products to support its management information needs.

Easing the Migration

The primary goal is to sunset the EPM solution. EPM customers that have any report that they need to continue to use must be migrated to another solution. When one is considering that thousands of reports are involved, this is a considerable workload to recreate each of these reports manually.

The Advanced Data Analytics Lab (ADAL) is a small team of Data Scientists, Developers, and Analysts that provide a variety of analytic and 'technology proof of concept' support within SSA. Outlined in this paper are technical methods as recommended by ADAL that SSA and federal agencies can use to leverage scripts and restful web services to help track and ease the administrative and development overhead of the migration process.

Analysis – Creating the Data

Components of an Enterprise Report Solution

Enterprise Report Solutions provide a framework for customers that typically consist of the following features, which enable one to:

- Create data connections to a variety of data sources, which can include databases, spreadsheets, or text files.
- View, retrieve, manipulate, filter, and sort the data.
- Display the data in a variety of ways in reports, tables, pivots, dashboards, or graphical views.
- Share reports and content with other users.
- Export the data to variety of formats including pdf or spreadsheets.
- Control who has access to the each of the features, reports, and content.

In order to provide this framework, most Enterprise Report Solution have the following components:

- Database to track details about the reports, data content, users, and user privileges.
- Web front end or desktop client application for users to access, create, edit and share reports.
- Web or other front end for administrators to maintain solution and grant user privileges.
- Restful web services or application programming interfaces (APIs) to enable administrators to perform maintenance activities on bulk set of objects such as moving reports from one virtual folder to another, uploading new content, or archiving outdated content.

EPM consists of a database, web front end, web administration tools and restful web services to provide the enterprise report features and this is true of other well-known enterprise report solutions such as WebFocus, Tableau, and SQL Server Reporting Services (SSRS). If you have access to the database, you can theoretically reconstruct much of the folder information, list of reports, and some of the content. Afterwards, you can pair this process with the restful web service APIs of the target solution to migrate the content.

When exploring and leveraging the solution database, it is best to work with a copy of the database and not the live one as you could unintentionally break the application if you change any data. Vendors usually prefer that customers leverage their technical staff for database related work through consulting or support hours. I worked with a snapshot of the database and not the live production database for this effort.

You can download a document on the database model from Oracle's Support Site at https://docs.oracle.com/cd/E40248_01/epm.1112/epm_data_models.zip. This link downloads a zipped file that contains several documents on the suite of Hyperion Products. The file pertinent to this analysis is the ReportingandAnalysis.pdf. The document provides a list of all the tables, columns, primary keys, foreign keys and what those keys link to which is helpful. It does not contain descriptions of the tables or columns so we really had to put on our investigative hats on and do quite a bit of data exploration to make sense of the database structure.

Fortunately, there is comprehensive documentation for the Restful Web Services Application programming interface APIs. Vendors provide these services to support administrative and developer needs and encourage their use. They also provide documentation, code samples, and training to customers.

The EPM Database

I have already cautioned you about using care when working with the database so be sure to work with a copy and not the live database. Office of Systems houses the production database on an Oracle server. I ported much of the data to a SQL Server database. One could use another Oracle Database, a MySQL database or even a NoSQL database if preferred. I picked SQL Server 2017 because of the string functions that made it much easier for to reconstruct the report content and reduced the amount of code I had to write. I will point out the portions of code that leverage those new functions where applicable.

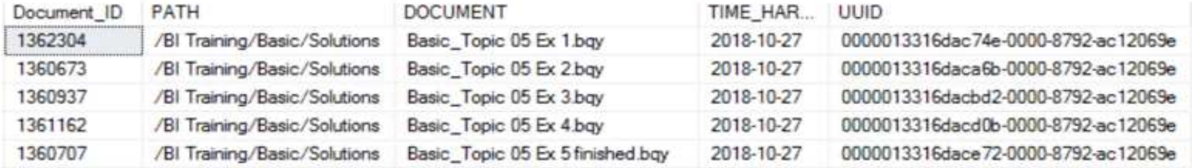
The EPM database has hundreds of tables. I extensively reviewed the tables and narrowed down the list of the most critical tables to this effort to fifty-five tables as shown in Appendix A. In EPM, you refer to all reports as documents. Each document contains one or more sections. The sections can be data models, queries, results, reports, pivots, charts, or dashboards. Sections can also be dependent on each other. For instance, one query may derive from another query, which may derive from a data model.

How many documents do you have?

The V8_CONTAINER table appears to be a master table within the database for all objects including but not limited to folders, documents, and open catalog extension (OCEs). OCEs are the objects that define the data source and connection information. Other enterprise report solutions use the term data source, data adaptor, or database connection instead of OCE.   The CONTAINER_UUID is the primary key for this table and contains the unique identifier for each object. The CONTAINER_UUID column connects to the foreign key UUID column in V8_OCE and V8_CONT_VERSION tables.

The V8_CONT_VERSION table uses both the CONTAINER_UUID and VERSION_NUMBER columns as a primary key and these columns are used together to connect to the matching foreign key columns in the V8_H_DOCUMENT, V8_QRY_DB_CONN, V8_BQ_SECTION, and V8_FOLDER tables. These tables are important tables that contain information about objects, data connections and virtual folders.

Let us take a deeper look at how document information is stored and how we can identify the document sections and content. The table V8_H_DOCUMENT contains a row for every document as shown in the image below. The V8_CONTAINER table will also have a row for every document but the V8_CONTAINER table contains unique rows for other objects such as folders and OCE rows in addition to documents. The V8_H_DOCUMENT only contains rows for all documents, not any other objects.



Document_ID	PATH	DOCUMENT	TIME_HAR...	UUID
1362304	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 1.bqy	2018-10-27	000013316dac74e-0000-8792-ac12069e
1360673	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 2.bqy	2018-10-27	000013316daca6b-0000-8792-ac12069e
1360937	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 3.bqy	2018-10-27	000013316dacbd2-0000-8792-ac12069e
1361162	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 4.bqy	2018-10-27	000013316dacd0b-0000-8792-ac12069e
1360707	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 5 finished.bqy	2018-10-27	000013316dace72-0000-8792-ac12069e

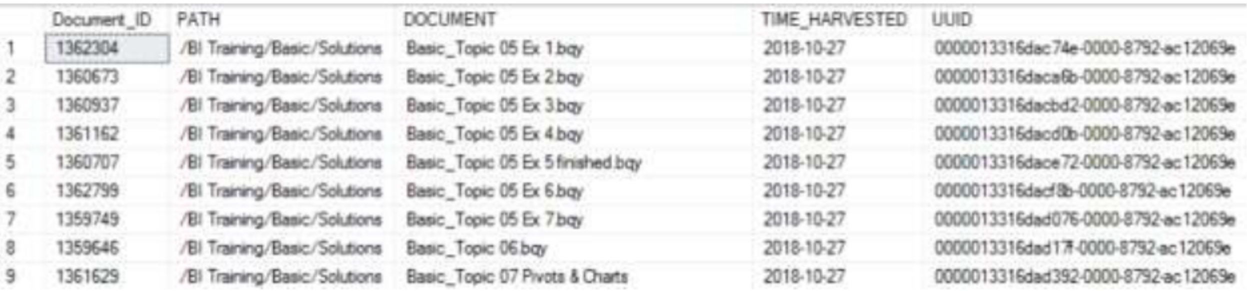
The DOCUMENT_ID column is the primary key column and uniquely identifies each document in the V8_H_DOCUMENT table. To get a count of how many documents we have I can run the query below which returns in this case, 57,665 rows.

```
SELECT count(*) from  V8_H_DOCUMENT
```

This does not account for test or old data that does not need to be migrated. If you only need to worry about certain folders, it is possible to look at documents within specific folders. For instance, if I want look at documents in the directory 'BI Training/Basic/Solutions', I can run the following query:

```
select Document_ID, PATH, DOCUMENT, TIME_HARVESTED, UUID from V8_H_DOCUMENT
  where path = '/BI Training/Basic/Solutions'
  order by Document
```

This returned fourteen rows. Here is an excerpt of the first nine of those fourteen rows:



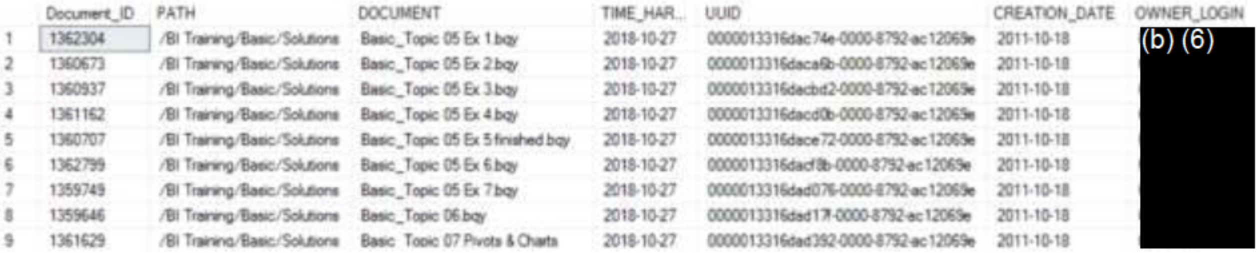
	Document_ID	PATH	DOCUMENT	TIME_HARVESTED	UUID
1	1362304	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 1.bqy	2018-10-27	0000013316dac74e-0000-8792-ac12069e
2	1360673	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 2.bqy	2018-10-27	0000013316daca6b-0000-8792-ac12069e
3	1360937	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 3.bqy	2018-10-27	0000013316dacbd2-0000-8792-ac12069e
4	1361162	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 4.bqy	2018-10-27	0000013316dacd0b-0000-8792-ac12069e
5	1360707	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 5 finished.bqy	2018-10-27	0000013316dace72-0000-8792-ac12069e
6	1362799	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 6.bqy	2018-10-27	0000013316dacf8b-0000-8792-ac12069e
7	1359749	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 7.bqy	2018-10-27	0000013316dad076-0000-8792-ac12069e
8	1359646	/BI Training/Basic/Solutions	Basic_Topic 06.bqy	2018-10-27	0000013316dad17f-0000-8792-ac12069e
9	1361629	/BI Training/Basic/Solutions	Basic_Topic 07 Pivots & Charts	2018-10-27	0000013316dad392-0000-8792-ac12069e

You can see from the results that for the first row, we have document with a DOCUMENT_ID of 1362304 and it has a name of 'Basic_Topic 05 EX 1.bqy', is located in the virtual folder called 'Solutions' and can be found by navigating through the page 'BI Training/Basic/Solutions'. It was last run on 10-27-2018.

The V8_H_DOCUMENT table is a great starting point to figuring out how many documents you have. Of course, you may also want to figure out who owns the document so that you can reach out to them. To get details on who owns the document and its creation date, you need use both the V8_CONTAINER and the V8_H_DOCUMENT tables. Therefore, we can build onto our first query to get the owner and creation date info:

```
  select d.Document_ID, d.PATH, d.DOCUMENT, d.TIME_HARVESTED, d.UUID,
  c.CREATION_DATE, c.OWNER_LOGIN
  from V8_H_DOCUMENT d
  inner join [V8_CONTAINER] c on d.UUID = c.CONTAINER_UUID
  where path = '/BI Training/Basic/Solutions'
  order by Document
```

The results with two additional columns of CREATION_DATE and OWNER_LOGIN (redacted):



	Document_ID	PATH	DOCUMENT	TIME_HAR...	UUID	CREATION_DATE	OWNER_LOGIN
1	1362304	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 1.bqy	2018-10-27	0000013316dac74e-0000-8792-ac12069e	2011-10-18	(b) (6) [redacted]
2	1360673	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 2.bqy	2018-10-27	0000013316daca6b-0000-8792-ac12069e	2011-10-18	[redacted]
3	1360937	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 3.bqy	2018-10-27	0000013316dacbd2-0000-8792-ac12069e	2011-10-18	[redacted]
4	1361162	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 4.bqy	2018-10-27	0000013316dacd0b-0000-8792-ac12069e	2011-10-18	[redacted]
5	1360707	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 5 finished.bqy	2018-10-27	0000013316dace72-0000-8792-ac12069e	2011-10-18	[redacted]
6	1362799	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 6.bqy	2018-10-27	0000013316dacf8b-0000-8792-ac12069e	2011-10-18	[redacted]
7	1359749	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 7.bqy	2018-10-27	0000013316dad076-0000-8792-ac12069e	2011-10-18	[redacted]
8	1359646	/BI Training/Basic/Solutions	Basic_Topic 06.bqy	2018-10-27	0000013316dad17f-0000-8792-ac12069e	2011-10-18	[redacted]
9	1361629	/BI Training/Basic/Solutions	Basic_Topic 07 Pivots & Charts	2018-10-27	0000013316dad392-0000-8792-ac12069e	2011-10-18	[redacted]

The OWNER_LOGIN is the network account of the user who logged into EPM and created the document.

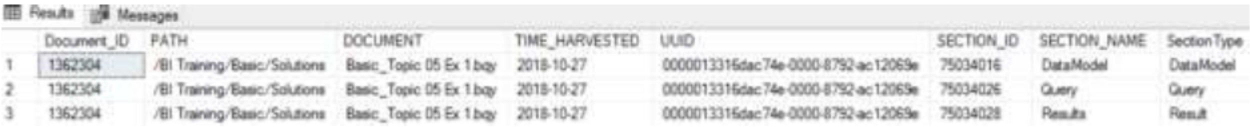
Section information for each document

It is straightforward from here to figure out what sections each document contains. To do this, we join the V8_H_DOCUMENT to the V8_H_SECTION and V8_H_SECTION_DEP tables. The V8_H_SECTION table contains a unique row for each section and uses the SECTION_ID field as the primary key for each section.

To look at all the sections for the Document with a DOCUMENT_ID of 1362304, we would use the following query:

```
select d.Document_ID, d.[PATH], d.DOCUMENT, d.TIME_HARVESTED, d.UUID,
 s.SECTION_ID, s.SECTION_NAME, cs.[NAME] as SectionType
from V8_H_DOCUMENT d
inner join [V8_H_SECTION] s on d.DOCUMENT_ID = s.DOCUMENT_ID
inner join [V8_H_CONSTANT] cs on cs.ID = s.SECTION_TYPE_ID
where d.DOCUMENT_ID = 1362304
```

There are different section types so I also joined V8_H_SECTION to the V8_H_CONSTANT table, which serves as a reference table for those values. The results:

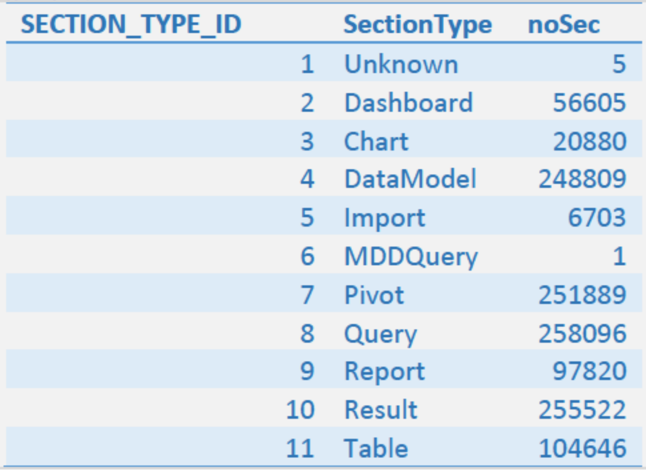


	Document_ID	PATH	DOCUMENT	TIME_HARVESTED	UUID	SECTION_ID	SECTION_NAME	SectionType
1	1362304	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 1.bqy	2018-10-27	00000133316dac74e-0000-8792-ac12069e	75034016	DataModel	DataModel
2	1362304	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 1.bqy	2018-10-27	00000133316dac74e-0000-8792-ac12069e	75034026	Query	Query
3	1362304	/BI Training/Basic/Solutions	Basic_Topic 05 Ex 1.bqy	2018-10-27	00000133316dac74e-0000-8792-ac12069e	75034028	Results	Result

The results show that Document 1362304 has three sections, a Data Model, Query, and Results.

We already know we have 57,665 documents. Now we know that each document consists of sections. The sections contain the logic to determine what data to use and the data display instructions. Let us see how many section types we have and how many of each type we have for all of our documents.

```
--count all sections for each document
select  s.SECTION_TYPE_ID,
c.NAME as SectionType, count(*) as noSec
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.DOCUMENT_ID = s.DOCUMENT_ID
inner join [V8_H_CONSTANT] c on s.SECTION_TYPE_ID = c.ID
group by s.SECTION_TYPE_ID, c.NAME
```



SECTION_TYPE_ID	SectionType	noSec
1	Unknown	5
2	Dashboard	56605
3	Chart	20880
4	DataModel	248809
5	Import	6703
6	MDDQuery	1
7	Pivot	251889
8	Query	258096
9	Report	97820
10	Result	255522
11	Table	104646

So 57,665 documents may not seem to be an astronomical number but when you look at the section counts, the numbers get bigger. There are 248,809 data model and 258,096 query sections, which hint at the effort needed to migrate that content.

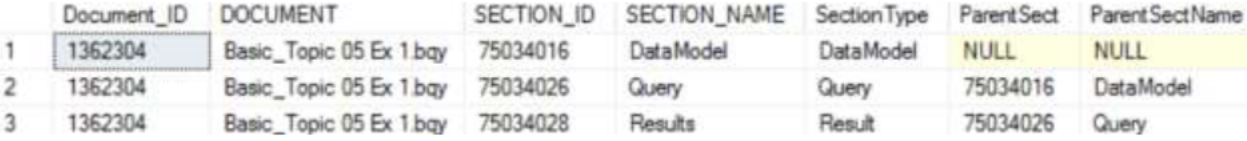
This count includes test and development content that may not need to be migrated so let us be conservative and estimate that for the data model sections, we are only going to migrate twenty percent. That is still 49,762 data model sections. If you consider it might take an analyst four to eight hours for each section at a labor cost of $80 per hour. That is a range of 199,048 to 398,096 hours at a cost of $15,923,840 to $31,847,680 for just the data model section.

Section Dependency

It is helpful to know if any sections are dependent on data of another section. When you first create a document in EPM, you associate it with an OCE and then one or more data models. EPM saves those database models as sections for each document. Then within EPM, you can add additional sections such as queries, results, reports, charts and dashboards that pull from those data models. You can nest the dependency for these sections as needed. For example, you could have one query section pull from a data model and then a second query section could use the data from the first query. Below I am tracing the dependency of document sections for document 1362304 using the following query:

```
   select d.Document_ID, d.DOCUMENT, s.SECTION_ID, s.SECTION_NAME,
         cs.[NAME] as SectionType, s2.SECTION_ID as ParentSect,
 s2.SECTION_NAME as ParentSectName
 from V8_H_DOCUMENT d
 inner join [V8_H_SECTION] s on d.DOCUMENT_ID = s.DOCUMENT_ID
 inner join [V8_H_CONSTANT] cs on cs.ID = s.SECTION_TYPE_ID
 left outer join [V8_H_SECTION_DEP] sd on s.SECTION_ID = sd.[CHILD_SECTION_ID]
 left outer join [V8_H_SECTION] s2  on sd.PARENT_SECTION_ID = s2.SECTION_ID
 where d.DOCUMENT_ID = 1362304
 order by s.SECTION_ID asc
```

The results from this query:



	Document_ID	DOCUMENT	SECTION_ID	SECTION_NAME	SectionType	ParentSect	ParentSectName
1	1362304	Basic_Topic 05 Ex 1.bqy	75034016	DataModel	DataModel	NULL	NULL
2	1362304	Basic_Topic 05 Ex 1.bqy	75034026	Query	Query	75034016	DataModel
3	1362304	Basic_Topic 05 Ex 1.bqy	75034028	Results	Result	75034026	Query

You can see that the Data Model is the parent section for the Query and the Query is the parent section for Result.

Now to tackle the content for each section. The goal here is to see if we can recreate the SQL queries to replicate the output of the sections so that we can take those queries and use them in another reporting tool to replace EPM. To do this, we will focus on the individual sections and start with identifying the tables, columns, filters, sort fields and computed items that EPM uses in each section to produce the output. EPM stores information on the tables, columns, rows, and other pertinent section information in groups of tables specific to each group. For instance, for the data model sections, the EPM stores the data model section in the following tables:

V8_H_DM_COLUMN
V8_H_DM_DIGEST
V8_H_DM_JOIN
V8_H_DM_LEAF_COL
V8_H_DM_LIMIT_COL

V8_H_DM_META_DEP
V8_H_DM_SECTION
V8_H_DM_TABLE

EPM stores data about the query sections in:

V8_H_QRY_COLUMN
V8_H_QRY_DM_REF
V8_H_QRY_QUERY
V8_H_QRYCOL_DMCOL
V8_H_QRYCOL_RSCOL

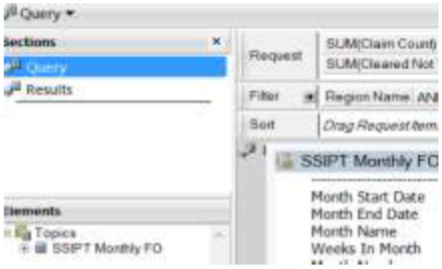
EPM stores the results section in:

V8_H_RS_COLUMN
V8_H_RS_COMP_ITEM
V8_H_RS_LIMIT_COL

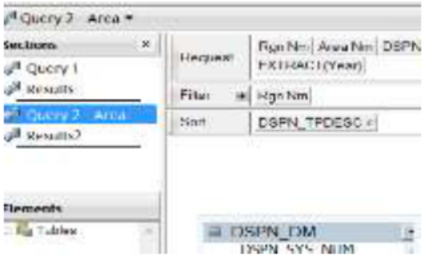
Data Model Section Type

In EPM, the data model tables are the tables that make available the list of columns for users to select from when building queries, results, and other sections. In EPM, when you review the document, you will see the individual sections show up in the sections pane on the left except for the data model sections. The data model exists as the list of elements that you can select to use in queries. If you have more than one data model, you will see that different sections are in groups associated with the different data models. When you select a query, the applicable fields for the data model show up in that group. The image below is a screenshot that illustrates this.

One Data Model



Two Data Models



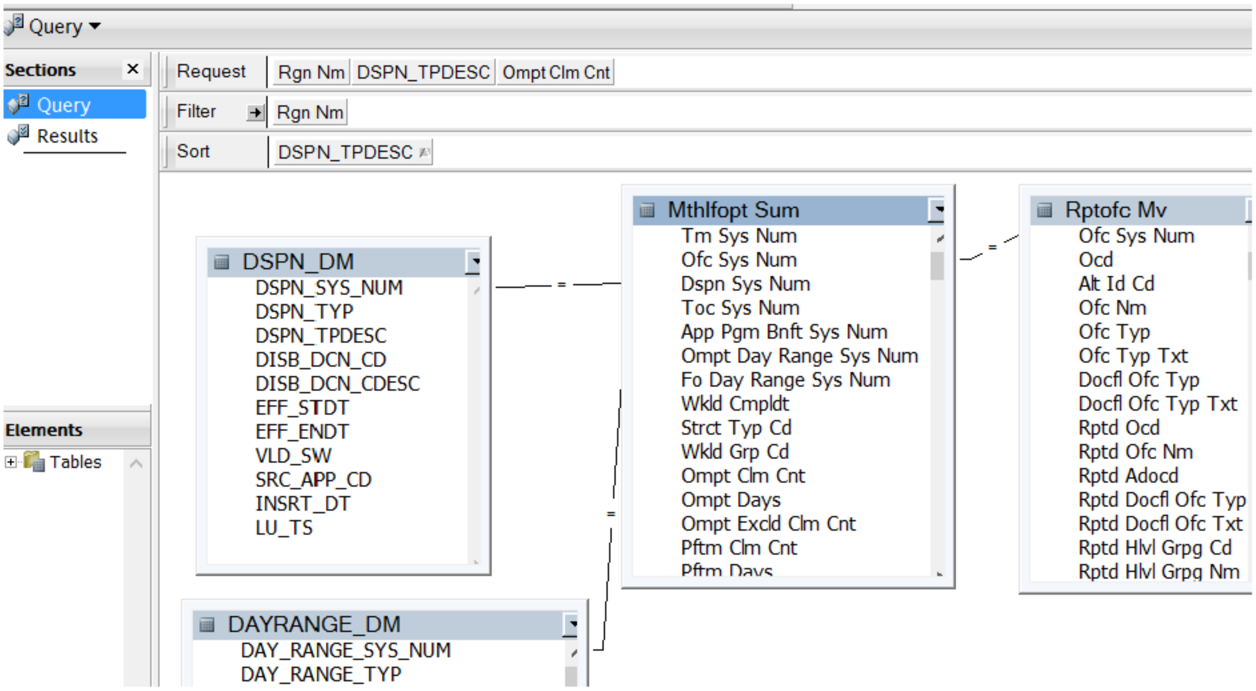
Notice how **Query 2- Area** falls below the black line indicating it uses a different model. Because in EPM, the data models serve as a source to select fields from other sections, you may or may not need to recreate them. It depends on the reporting tool and if you need to stage data for users to access as a starting point so that they have a controlled group of data to select for reports.

There are two categories of data models in EPM, which are Tables and Topics. If a data model pulls data directly from a data source it falls in the Table category. If it derives from an internal EPM view of data from other data models, it falls in the Topics category. Data models that fall in the Tables category are easier to recreate and I will focus on that first.

Data Model – Table Category

To recreate a SQL query for a data model, you use the V8_H_DM_Table and V8_H_DM_COLUMN to identify the tables and columns used by each section. A warning though, the columns and tables listed in these tables may have friendly names or alias that include spaces or other characters. Column and table names in a physical database should not have spaces or columns and in the circumstances where they do, you have to include square brackets around the name or the SQL query will not execute. Later you will see in some of the recreation scripts, that I add brackets to the table and column names to mitigate this issue where I am able. The actual physical table and column names are stored in the V8_H_DB_Table and V_H_DB_Column tables. You can join the data model table to the physical tables through the foreign key columns RDBMS_TABLE_ID and RDBMS_COLUMN_ID.

I have a sample document that I am going to work with to trace the data model. My document is called SSIPT Simple Query 1 and is located in the virtual EPM folder of '/BI Central Office Folders/DCBFM/DCBFM FPA/Development/migrationScript_test'. Here is what you see when you look at this document in the EPM client application:

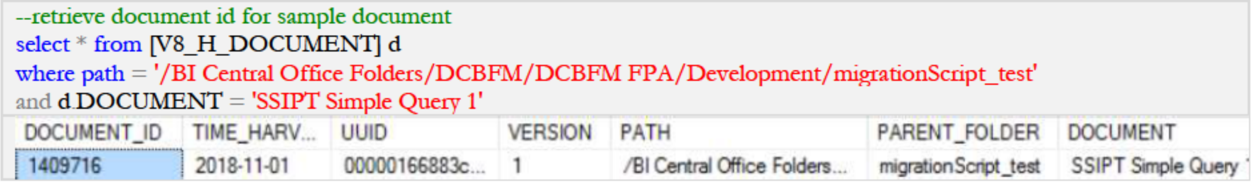


Request: Rgn Nm | DSPN_TPDESC | Ompt Clm Cnt

Filter: Rgn Nm

Sort: DSPN_TPDESC

Within the database, we need to figure out what the document id is and we can do that with the following query.

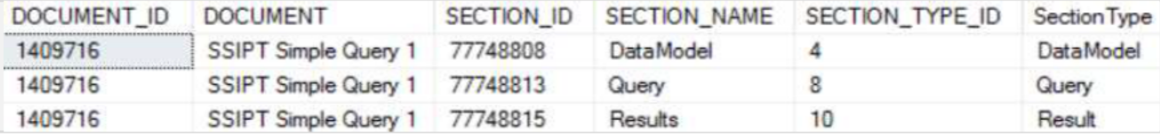


```
--retrieve document id for sample document
select * from [V8_H_DOCUMENT] d
where path = '/BI Central Office Folders/DCBFM/DCBFM FPA/Development/migrationScript_test'
and d.DOCUMENT = 'SSIPT Simple Query 1'
```

DOCUMENT_ID	TIME_HARV...	UUID	VERSION	PATH	PARENT_FOLDER	DOCUMENT
1409716	2018-11-01	00000166883c...	1	/BI Central Office Folders...	migrationScript_test	SSIPT Simple Query 1

The output shows a document ID of 1409716. Now I need to see the list of sections and get the applicable section_id for the data model.

```
select d.DOCUMENT_ID, d.DOCUMENT, s.SECTION_ID, s.SECTION_NAME, s.SECTION_TYPE_ID,
c.NAME as SectionType
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.DOCUMENT_ID = s.DOCUMENT_ID
inner join [V8_H_CONSTANT] c on s.SECTION_TYPE_ID = c.ID
where d.DOCUMENT_ID = 1409716
```

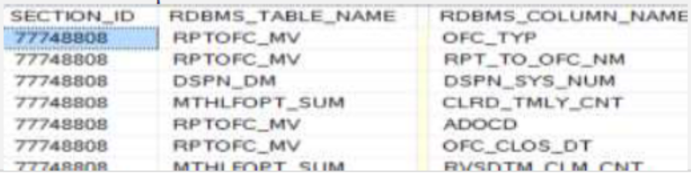


DOCUMENT_ID	DOCUMENT	SECTION_ID	SECTION_NAME	SECTION_TYPE_ID	SectionType
1409716	SSIPT Simple Query 1	77748808	DataModel	4	DataModel
1409716	SSIPT Simple Query 1	77748813	Query	8	Query
1409716	SSIPT Simple Query 1	77748815	Results	10	Result

This shows that there is one data model that I can identify with a SECTION_ID of 77748808. That is all I need to run a query to get the tables and columns for this data model.

```
--This query returns the display fields and their respective tables in the data model
select distinct  s.SECTION_ID, dbt.RDBMS_TABLE_NAME, dmc1.DEFINITION, dbc1.RDBMS_COLUMN_NAME
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join  [V8_H_DM_TABLE] dmt  on s.SECTION_ID = dmt.SECTION_ID
inner join [V8_H_DB_TABLE] dbt on dmt.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
inner join  [V8_H_DM_COLUMN] dmc1  on dmt.DM_TABLE_ID =dmc1.DM_TABLE_ID
inner join  [V8_H_DB_COLUMN] dbc1 on dmc1.RDBMS_COLUMN_ID = dbc1.RDBMS_COLUMN_ID
 where SECTION_TYPE_ID = 4 and s.section_id = 77748808
```

Part of the output:



SECTION_ID	RDBMS_TABLE_NAME	RDBMS_COLUMN_NAME
77748808	RPTOFC_MV	OFC_TYP
77748808	RPTOFC_MV	RPT_TO_OFC_NM
77748808	DSPN_DM	DSPN_SYS_NUM
77748808	MTHLFOPT_SUM	CLRD_TMLY_CNT
77748808	RPTOFC_MV	ADOCD
77748808	RPTOFC_MV	OFC_CLOS_DT
77748808	MTHLFOPT_SUM	RVSDTM_CLM_CNT

Next, I am going to build on this query to individually return the display columns, tables, table joins, and filters and then format each of those results by rolling them up to one row for each data model section. The image shows seven column names for section_id 77748808.

The typical format of a SQL Query is **'SELECT <table1.column1, table1.column2 > FROM <table1, table2> WHERE <table1.col1 = table2.col1> AND <col1 = filter value>.'** The columns shown in the image above are the columns I need to use in the SELECT statement. If I take the first two columns, here is how I need them to appear when part of a SELECT statement: 'SELECT RPTOFC_MV.OFC_TYP, RPTOFC_MV.RPT_TO_OFC_NM FROM RPTOFC_MV.' To achieve this, I need to prefix each column with the table name followed by a period, add a comma to the end and roll all of the columns for each section to one row. As I mentioned earlier, SQL 2017 has string functions that make it possible for me to do this.

I use the SQL Server 2017 CONCAT and STRING_AGG functions. The CONCAT function combines fields together into one column with additional characters as needed. The STRING_AGG function rolls the combined columns into one big column for each section, and separates each of the combined columns within that section with a comma. The STRING_AGG knows to groups all of this by section because I specified SECTION_ID in the GROUP BY parameter.

If I did not specify a 'GROUP BY SECTION_ID', it would actually merge all of the results for all sections into one row.

Data Models usually have many display fields and if you look at the output, it matches the fields the users can potentially select from to build their queries.

```
--Data model display fields --format for query
 select x1.section_id,
 string_agg(cast( concat('[',x1.RDBMS_TABLE_NAME,'].[',x1.RDBMS_COLUMN_NAME,']') as varchar(max)), ', ') as
DisplayFields
 from (
 select distinct  s.SECTION_ID, dbt.RDBMS_TABLE_NAME, dbc1.RDBMS_COLUMN_NAME
 from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join  [V8_H_DM_TABLE] dmt  on s.SECTION_ID = dmt.SECTION_ID
inner join [V8_H_DB_TABLE] dbt on dmt.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
inner join  [V8_H_DM_COLUMN] dmc1  on dmt.DM_TABLE_ID =dmc1.DM_TABLE_ID
inner join  [V8_H_DB_COLUMN] dbc1 on dmc1.RDBMS_COLUMN_ID = dbc1.RDBMS_COLUMN_ID
 where SECTION_TYPE_ID = 4 and s.section_id = 77748808
 ) as x1
group by x1.SECTION_ID
```

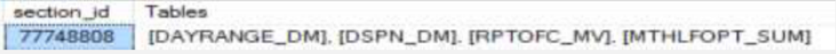
```
section_id  DisplayFields
77748808    [MTHLFOPT_SUM].[CLRD_TMLY_CNT], [MTHLFOPT_SUM].[PFTM_CLM_CNT], [RPTOFC_MV].[RPT_TO_OCD], [MTHLFOPT_SUM].[CLRD_EXCLD_CNT], [M...
```

Text View of results:

```
[MTHLFOPT_SUM].[CLRD_TMLY_CNT], [MTHLFOPT_SUM].[PFTM_CLM_CNT],
[RPTOFC_MV].[RPT_TO_OCD], [MTHLFOPT_SUM].[CLRD_EXCLD_CNT],
[MTHLFOPT_SUM].[DSPN_SYS_NUM], [MTHLFOPT_SUM].[LU_TS],
[MTHLFOPT_SUM].[TRNSTTM_1_CLM_CNT], [MTHLFOPT_SUM].[TRNSTTM_1_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[DDSTM_EXCLD_CLM_CNT], [MTHLFOPT_SUM].[FO1TM_CLM_CNT],
[MTHLFOPT_SUM].[FO2TM_DAYS], [RPTOFC_MV].[EFF_ENDT], [RPTOFC_MV].[ROCD],
[RPTOFC_MV].[RPTD_OCD], [DSPN_DM].[EFF_ENDT],
[MTHLFOPT_SUM].[FO2TM_EXCLD_CLM_CNT], [MTHLFOPT_SUM].[PFTM_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[TRNSTTM_3_EXCLD_CLM_CNT], [RPTOFC_MV].[OFC_REOPN_DT],
[DAYRANGE_DM].[EFF_STDT], [DSPN_DM].[SRC_APP_CD], [DAYRANGE_DM].[DAY_RANGE_CDESC],
[DAYRANGE_DM].[SRC_APP_CD], [MTHLFOPT_SUM].[OMPT_DAY_RANGE_SYS_NUM],
[MTHLFOPT_SUM].[RVSDTM_CLM_CNT], [MTHLFOPT_SUM].[TRNSTTM_1_DAYS],
[RPTOFC_MV].[RPT_TO_OFC_TYP_NM], [DAYRANGE_DM].[DAY_RANGE_SYS_NUM],
[DAYRANGE_DM].[DAY_RANGE_TYP], [DAYRANGE_DM].[INSRT_TS], [DAYRANGE_DM].[LU_TS],
[MTHLFOPT_SUM].[TM_SYS_NUM], [RPTOFC_MV].[ALT_ID_CD], [RPTOFC_MV].[OFC_NM],
[RPTOFC_MV].[OFC_TYP_TXT], [RPTOFC_MV].[VLD_SW], [MTHLFOPT_SUM].[FO2TM_CLM_CNT],
[MTHLFOPT_SUM].[PFTM_DAYS], [MTHLFOPT_SUM].[WKLD_CMPLDT], [RPTOFC_MV].[INSRT_TS],
[RPTOFC_MV].[MOD_NUM], [RPTOFC_MV].[ROLTR], [RPTOFC_MV].[RPT_TO_OFC_TYP],
[RPTOFC_MV].[RPTD_HLVL_GRPG_CD], [RPTOFC_MV].[RPTD_RI], [RPTOFC_MV].[RPTD_RSN_CD],
[DAYRANGE_DM].[DAY_RANGE_CD], [DAYRANGE_DM].[HIGH_CNT], [DSPN_DM].[DISB_DCN_CD],
[DSPN_DM].[LU_TS], [DSPN_DM].[VLD_SW], [MTHLFOPT_SUM].[OMPT_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[TRNSTTM_2_CLM_CNT], [RPTOFC_MV].[EFF_STDT], [RPTOFC_MV].[OFC_TYP],
[RPTOFC_MV].[RGN_ACR], [DSPN_DM].[DSPN_SYS_NUM], [MTHLFOPT_SUM].[DDSTM_DAYS],
[RPTOFC_MV].[ADOCD], [RPTOFC_MV].[ST_AGY_CD], [MTHLFOPT_SUM].[FOTM_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[TRNSTTM_2_DAYS], [MTHLFOPT_SUM].[TRNSTTM_2_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[TRNSTTM_3_CLM_CNT], [RPTOFC_MV].[SRC_APP_CD], [DSPN_DM].[INSRT_DT],
[MTHLFOPT_SUM].[MED_FLDR_ORECTYP_SYS_NUM], [RPTOFC_MV].[RPT_TO_OFC_NM],
[RPTOFC_MV].[RPTD_OFC_NM], [DAYRANGE_DM].[EFF_END], [DAYRANGE_DM].[LOW_CNT],
[MTHLFOPT_SUM].[CLRD_NOT_TMLY_CNT], [MTHLFOPT_SUM].[DQBTM_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[OMPT_DAYS], [MTHLFOPT_SUM].[STRCT_TYP_CD], [RPTOFC_MV].[AREA_NM],
[RPTOFC_MV].[OCD], [RPTOFC_MV].[OFC_CLOS_DT], [RPTOFC_MV].[OFC_ESTB_DT],
[RPTOFC_MV].[RPTD_RSN_DESC], [RPTOFC_MV].[ST_ABBR],
[MTHLFOPT_SUM].[APP_PGM_BNFT_SYS_NUM], [MTHLFOPT_SUM].[FO1TM_DAYS],
[MTHLFOPT_SUM].[OFC_SYS_NUM], [RPTOFC_MV].[RPTD_DOCFL_OFC_TYP],
[DSPN_DM].[DISB_DCN_CDESC], [MTHLFOPT_SUM].[FOTM_CLM_CNT],
[MTHLFOPT_SUM].[INSRT_TS], [MTHLFOPT_SUM].[RVSDTM_EXCLD_CLM_CNT],
[DSPN_DM].[DSPN_TPDESC], [MTHLFOPT_SUM].[TRNSTTM_3_DAYS],
[RPTOFC_MV].[DOCFL_OFC_TYP], [RPTOFC_MV].[RPTD_ADOCD], [DAYRANGE_DM].[VLD_SW],
[DSPN_DM].[DSPN_TYP], [RPTOFC_MV].[RGN_NM], [DSPN_DM].[EFF_STDT],
[MTHLFOPT_SUM].[DDSTM_CLM_CNT], [MTHLFOPT_SUM].[FO_DAY_RANGE_SYS_NUM],
[MTHLFOPT_SUM].[RVSDTM_DAYS], [MTHLFOPT_SUM].[TOC_SYS_NUM],
[MTHLFOPT_SUM].[WKLD_GRP_CD], [RPTOFC_MV].[DOCFL_OFC_TYP_TXT],
[RPTOFC_MV].[OFC_SYS_NUM], [RPTOFC_MV].[RPTD_DOCFL_OFC_TXT], [RPTOFC_MV].[ST],
[MTHLFOPT_SUM].[DQBTM_DAYS], [MTHLFOPT_SUM].[FOTM_DAYS],
[MTHLFOPT_SUM].[OMPT_CLM_CNT], [RPTOFC_MV].[STNM],
[MTHLFOPT_SUM].[DQBTM_CLM_CNT], [MTHLFOPT_SUM].[FO1TM_EXCLD_CLM_CNT],
[RPTOFC_MV].[LU_TS], [RPTOFC_MV].[RPTD_HLVL_GRPG_NM]
```

Tables:

```
--data model retrieve tables
 select distinct dmt.section_id, string_agg(cast(concat('[',dbt.RDBMS_TABLE_NAME,']') as varchar(max)), ', ') as Tables
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_DM_TABLE] dmt on s.SECTION_ID = dmt.SECTION_ID
inner join [V8_H_DB_TABLE] dbt on dmt.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
where SECTION_TYPE_ID = 4 and s.section_id = 77748808
group by dmt.section_id
```



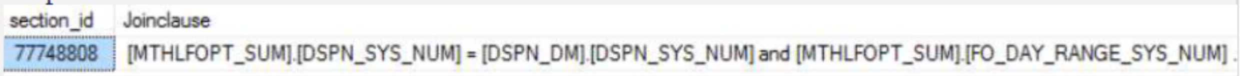
```
section_id  Tables
77748808    [DAYRANGE_DM], [DSPN_DM], [RPTOFC_MV], [MTHLFOPT_SUM]
```

Text: [DAYRANGE_DM], [DSPN_DM], [RPTOFC_MV], [MTHLFOPT_SUM]

Table Joins:

```
--Data model retrieve table joins
select z.section_id,
string_agg(cast(concat('[',z.tbl1,']','.','[',z.col1,']', ' = ', '[',z.tbl2,']','.','[', z.col2,']') as varchar(max)),' and ') as Joinclause
from (
select distinct s.section_id, dbt1.RDBMS_TABLE_NAME as tbl1, dbc1.RDBMS_COLUMN_NAME as col1,
dbt2.RDBMS_TABLE_NAME as tbl2, dbc2.RDBMS_COLUMN_NAME as col2
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_DM_JOIN] dmj on s.SECTION_ID = dmj.SECTION_ID
inner join [V8_H_DM_COLUMN] dmc1 on dmj.FROM_HPSU_COLUMN_ID = dmc1.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc1 on dmc1.RDBMS_COLUMN_ID = dbc1.RDBMS_COLUMN_ID
inner join [V8_H_DB_TABLE] dbt1 on dbc1.RDBMS_TABLE_ID = dbt1.RDBMS_TABLE_ID
inner join [V8_H_DM_COLUMN] dmc2 on dmj.TO_HPSU_COLUMN_ID = dmc2.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc2 on dmc2.RDBMS_COLUMN_ID = dbc2.RDBMS_COLUMN_ID
inner join  [V8_H_DB_TABLE] dbt2 on dbc2.RDBMS_TABLE_ID = dbt2.RDBMS_TABLE_ID
where SECTION_TYPE_ID = 4 and s.SECTION_ID = 77748808
) as z
group by z.SECTION_ID
```

Output:



```
section_id  Joinclause
77748808    [MTHLFOPT_SUM].[DSPN_SYS_NUM] = [DSPN_DM].[DSPN_SYS_NUM] and [MTHLFOPT_SUM].[FO_DAY_RANGE_SYS_NUM] .
```

TEXT:

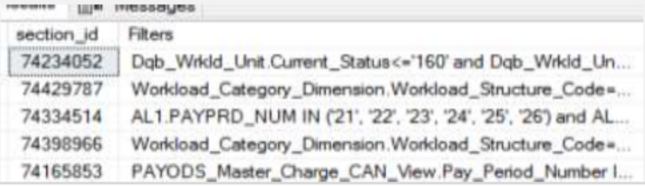
```
[MTHLFOPT_SUM].[DSPN_SYS_NUM] = [DSPN_DM].[DSPN_SYS_NUM] and
[MTHLFOPT_SUM].[FO_DAY_RANGE_SYS_NUM] = [DAYRANGE_DM].[DAY_RANGE_SYS_NUM] and
[MTHLFOPT_SUM].[OFC_SYS_NUM] = [RPTOFC_MV].[OFC_SYS_NUM]
```

To find out the filters that are used:

```
select s.section_id, cast(string_agg( dl.LIMIT_VALUE,' and ') as varchar(max)) as Filters
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_DM_LIMIT_COL] dl on dl.PARENT_SECTION_ID = s.SECTION_ID
where SECTION_TYPE_ID = 4 and s.SECTION_ID = 77748808
group by s.section_id
```

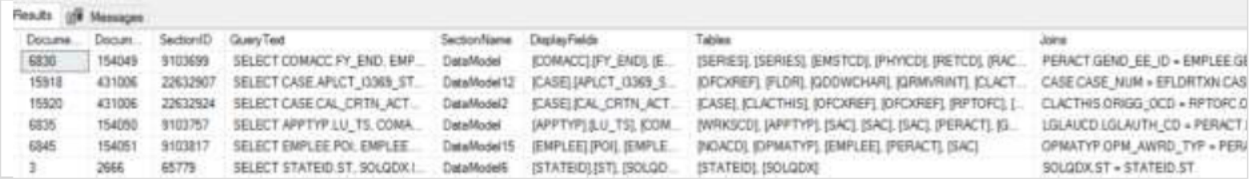
section_id	Filters

This particular data model does not have any filters but here are examples of results of some data model sections that do:



section_id	Filters
74234052	Dqb_Wrkld_Unit.Current_Status<='160' and Dqb_Wrkld_Un...
74429787	Workload_Category_Dimension.Workload_Structure_Code=,...
74334514	AL1.PAYPRD_NUM IN ('21', '22', '23', '24', '25', '26') and AL...
74398966	Workload_Category_Dimension.Workload_Structure_Code=...
74165853	PAYODS_Master_Charge_CAN_View.Pay_Period_Number I...

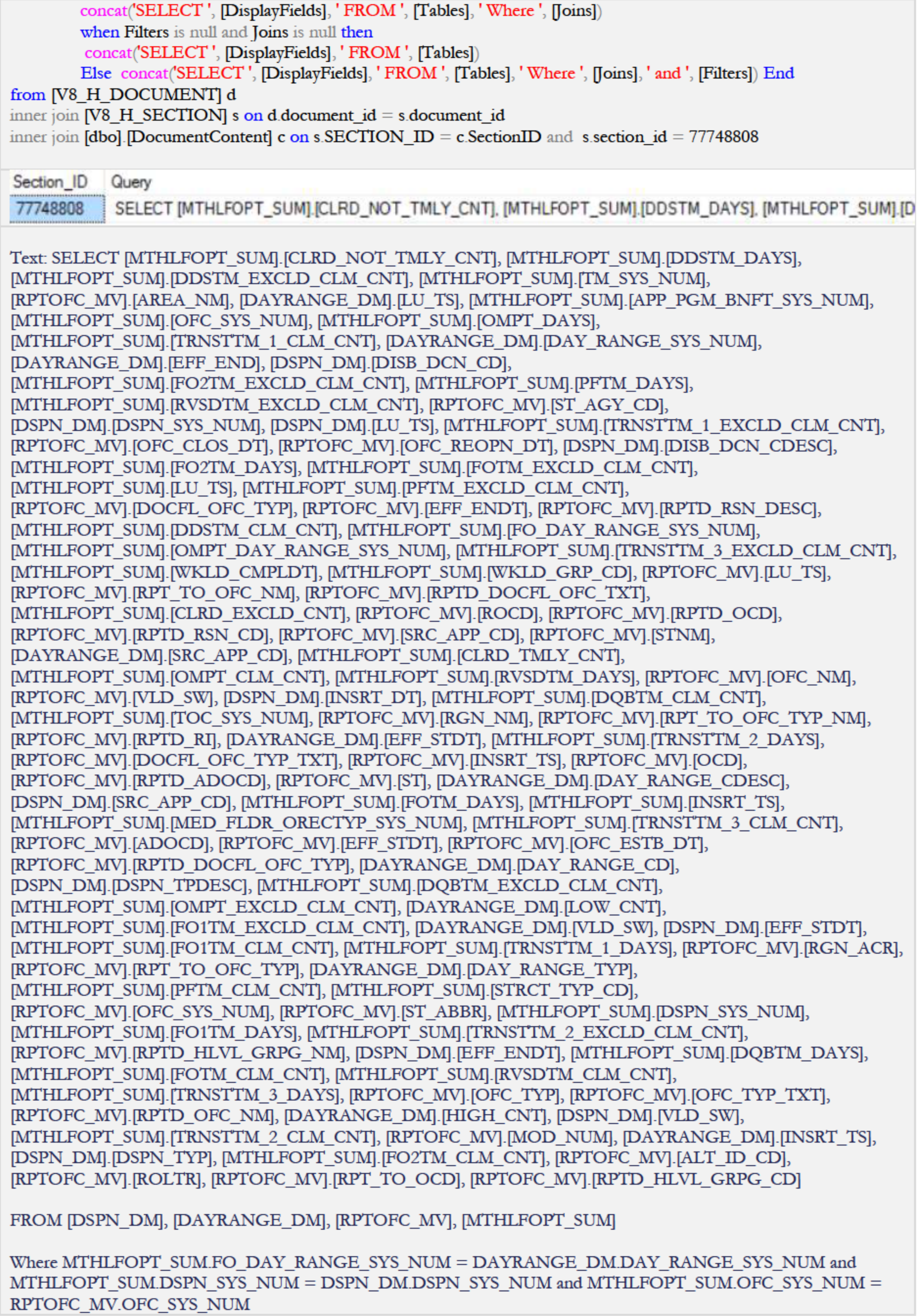
Data model sections do not contain computed or sort columns so we have everything we need to recreate the SQL Query by combining the results of the display fields and tables, the joins, and the filters. The V8_H_DM_COLUMN table contains 36,121,433 rows so performance is a consideration. Identifying the section metadata and dynamically recreating report content on the fly can slow performance when using a web front end or running a migration script. We can de-normalize data to improve application performance and this is acceptable as we are pulling data from EPM for management information purposes. This process does not maintain the integrity of the source data, which falls to the EPM application. In this instance, I opted to stage the individual section and related component information into a table called DocumentContent. I inserted a row into this table for each section and created individual columns to hold the rolled-up, formatted versions of the display fields, tables, joins, and filter. The code to create this table and insert the individual section information is contained in Appendix B. Then I use a final script to join the pieces together. Below is a screen shot of a sample of the data model information contained in the DocumentContent table.



Docume...	Docum...	SectionID	QueryText	SectionName	DisplayFields	Tables	Joins
6830	154049	9103699	SELECT COMACC.FY_END, EMP...	DataModel	[COMACC].[FY_END], [E...	[SERIES], [SERIES], [EMSTCD], [PHYICD], [RETCD], [RAC...	PERACT.GEND_EE_ID = EMPLEE.GE
15918	431006	22632907	SELECT CASE.APLCT_I3369_ST...	DataModel12	[CASE].[APLCT_I3369_S...	[OFCXREF], [FLDR], [QDOWCHAR], [QRMVRINT], [CLACT...	CASE.CASE_NUM = EFLDRTXN.CAS
15920	431006	22632924	SELECT CASE.CAL_CRTN_ACT...	DataModel2	[CASE].[CAL_CRTN_ACT...	[CASE], [CLACTHIS], [OFCXREF], [OFCXREF], [RPTOFC], [...	CLACTHIS.ORIGG_OCD = RPTOFC.O
6835	154050	9103757	SELECT APPTYP.LU_TS, COMA...	DataModel	[APPTYP].[LU_TS], [COM...	[WRKSCD], [APPTYP], [SAC], [SAC], [SAC], [PERACT], [G...	LGLAUCD.LGLAUTH_CD = PERACT.
6845	154051	9103817	SELECT EMPLEE.POI, EMPLEE...	DataModel15	[EMPLEE].[POI], [EMPLE...	[NOACD], [OPMATYP], [EMPLEE], [PERACT], [SAC]	OPMATYP.OPM_AWRD_TYP = PERA
3	2666	65779	SELECT STATEID.ST, SOLQDX.I...	DataModel6	[STATEID].[ST], [SOLQD...	[STATEID], [SOLQDX]	SOLQDX.ST = STATEID.ST

Now that you have staged all the data, you can put it all together to recreate your data model.

```
--query to put everything together for data model
select  s.[Section_ID],
'Query' = case when DisplayFields is null then ' no data' --tables joins filters
            when not Filters is null and not Joins is null then
            concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins], ' and ', [Filters])
            when not Filters is null and Joins is null and SortFields is null then
            concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Filters])
            when  Filters is null and not Joins is null then
```



```
            concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins])
            when Filters is null and Joins is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables])
            Else  concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins], ' and ', [Filters]) End
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [dbo].[DocumentContent] c on s.SECTION_ID = c.SectionID and  s.section_id = 77748808
```

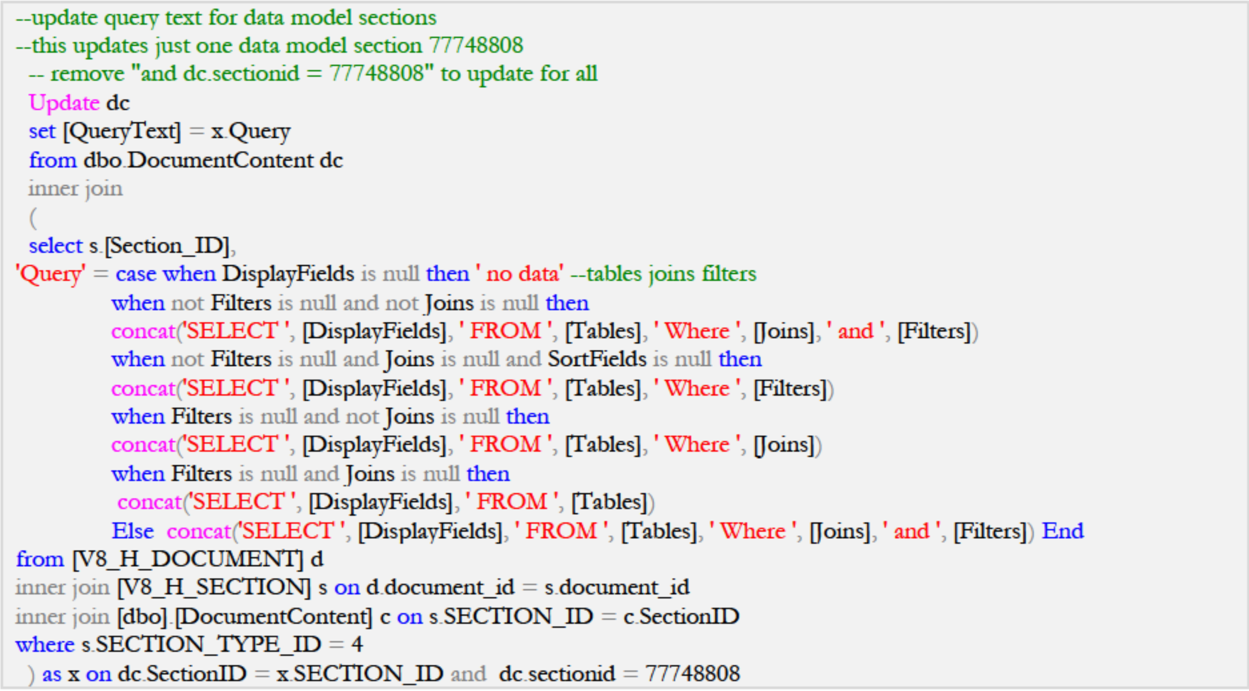
Section_ID	Query
77748808	SELECT [MTHLFOPT_SUM].[CLRD_NOT_TMLY_CNT], [MTHLFOPT_SUM].[DDSTM_DAYS], [MTHLFOPT_SUM].[D

```
Text: SELECT [MTHLFOPT_SUM].[CLRD_NOT_TMLY_CNT], [MTHLFOPT_SUM].[DDSTM_DAYS],
[MTHLFOPT_SUM].[DDSTM_EXCLD_CLM_CNT], [MTHLFOPT_SUM].[TM_SYS_NUM],
[RPTOFC_MV].[AREA_NM], [DAYRANGE_DM].[LU_TS], [MTHLFOPT_SUM].[APP_PGM_BNFT_SYS_NUM],
[MTHLFOPT_SUM].[OFC_SYS_NUM], [MTHLFOPT_SUM].[OMPT_DAYS],
[MTHLFOPT_SUM].[TRNSTTM_1_CLM_CNT], [DAYRANGE_DM].[DAY_RANGE_SYS_NUM],
[DAYRANGE_DM].[EFF_END], [DSPN_DM].[DISB_DCN_CD],
[MTHLFOPT_SUM].[FO2TM_EXCLD_CLM_CNT], [MTHLFOPT_SUM].[PFTM_DAYS],
[MTHLFOPT_SUM].[RVSDTM_EXCLD_CLM_CNT], [RPTOFC_MV].[ST_AGY_CD],
[DSPN_DM].[DSPN_SYS_NUM], [DSPN_DM].[LU_TS], [MTHLFOPT_SUM].[TRNSTTM_1_EXCLD_CLM_CNT],
[RPTOFC_MV].[OFC_CLOS_DT], [RPTOFC_MV].[OFC_REOPN_DT], [DSPN_DM].[DISB_DCN_CDESC],
[MTHLFOPT_SUM].[FO2TM_DAYS], [MTHLFOPT_SUM].[FOTM_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[LU_TS], [MTHLFOPT_SUM].[PFTM_EXCLD_CLM_CNT],
[RPTOFC_MV].[DOCFL_OFC_TYP], [RPTOFC_MV].[EFF_ENDT], [RPTOFC_MV].[RPTD_RSN_DESC],
[MTHLFOPT_SUM].[DDSTM_CLM_CNT], [MTHLFOPT_SUM].[FO_DAY_RANGE_SYS_NUM],
[MTHLFOPT_SUM].[OMPT_DAY_RANGE_SYS_NUM], [MTHLFOPT_SUM].[TRNSTTM_3_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[WKLD_CMPLDT], [MTHLFOPT_SUM].[WKLD_GRP_CD], [RPTOFC_MV].[LU_TS],
[RPTOFC_MV].[RPT_TO_OFC_NM], [RPTOFC_MV].[RPTD_DOCFL_OFC_TXT],
[MTHLFOPT_SUM].[CLRD_EXCLD_CNT], [RPTOFC_MV].[ROCD], [RPTOFC_MV].[RPTD_OCD],
[RPTOFC_MV].[RPTD_RSN_CD], [RPTOFC_MV].[SRC_APP_CD], [RPTOFC_MV].[STNM],
[DAYRANGE_DM].[SRC_APP_CD], [MTHLFOPT_SUM].[CLRD_TMLY_CNT],
[MTHLFOPT_SUM].[OMPT_CLM_CNT], [MTHLFOPT_SUM].[RVSDTM_DAYS], [RPTOFC_MV].[OFC_NM],
[RPTOFC_MV].[VLD_SW], [DSPN_DM].[INSRT_DT], [MTHLFOPT_SUM].[DQBTM_CLM_CNT],
[MTHLFOPT_SUM].[TOC_SYS_NUM], [RPTOFC_MV].[RGN_NM], [RPTOFC_MV].[RPT_TO_OFC_TYP_NM],
[RPTOFC_MV].[RPTD_RI], [DAYRANGE_DM].[EFF_STDT], [MTHLFOPT_SUM].[TRNSTTM_2_DAYS],
[RPTOFC_MV].[DOCFL_OFC_TYP_TXT], [RPTOFC_MV].[INSRT_TS], [RPTOFC_MV].[OCD],
[RPTOFC_MV].[RPTD_ADOCD], [RPTOFC_MV].[ST], [DAYRANGE_DM].[DAY_RANGE_CDESC],
[DSPN_DM].[SRC_APP_CD], [MTHLFOPT_SUM].[FOTM_DAYS], [MTHLFOPT_SUM].[INSRT_TS],
[MTHLFOPT_SUM].[MED_FLDR_ORECTYP_SYS_NUM], [MTHLFOPT_SUM].[TRNSTTM_3_CLM_CNT],
[RPTOFC_MV].[ADOCD], [RPTOFC_MV].[EFF_STDT], [RPTOFC_MV].[OFC_ESTB_DT],
[RPTOFC_MV].[RPTD_DOCFL_OFC_TYP], [DAYRANGE_DM].[DAY_RANGE_CD],
[DSPN_DM].[DSPN_TPDESC], [MTHLFOPT_SUM].[DQBTM_EXCLD_CLM_CNT],
[MTHLFOPT_SUM].[OMPT_EXCLD_CLM_CNT], [DAYRANGE_DM].[LOW_CNT],
[MTHLFOPT_SUM].[FO1TM_EXCLD_CLM_CNT], [DAYRANGE_DM].[VLD_SW], [DSPN_DM].[EFF_STDT],
[MTHLFOPT_SUM].[FO1TM_CLM_CNT], [MTHLFOPT_SUM].[TRNSTTM_1_DAYS], [RPTOFC_MV].[RGN_ACR],
[RPTOFC_MV].[RPT_TO_OFC_TYP], [DAYRANGE_DM].[DAY_RANGE_TYP],
[MTHLFOPT_SUM].[PFTM_CLM_CNT], [MTHLFOPT_SUM].[STRCT_TYP_CD],
[RPTOFC_MV].[OFC_SYS_NUM], [RPTOFC_MV].[ST_ABBR], [MTHLFOPT_SUM].[DSPN_SYS_NUM],
[MTHLFOPT_SUM].[FO1TM_DAYS], [MTHLFOPT_SUM].[TRNSTTM_2_EXCLD_CLM_CNT],
[RPTOFC_MV].[RPTD_HLVL_GRPG_NM], [DSPN_DM].[EFF_ENDT], [MTHLFOPT_SUM].[DQBTM_DAYS],
[MTHLFOPT_SUM].[FOTM_CLM_CNT], [MTHLFOPT_SUM].[RVSDTM_CLM_CNT],
[MTHLFOPT_SUM].[TRNSTTM_3_DAYS], [RPTOFC_MV].[OFC_TYP], [RPTOFC_MV].[OFC_TYP_TXT],
[RPTOFC_MV].[RPTD_OFC_NM], [DAYRANGE_DM].[HIGH_CNT], [DSPN_DM].[VLD_SW],
[MTHLFOPT_SUM].[TRNSTTM_2_CLM_CNT], [RPTOFC_MV].[MOD_NUM], [DAYRANGE_DM].[INSRT_TS],
[DSPN_DM].[DSPN_TYP], [MTHLFOPT_SUM].[FO2TM_CLM_CNT], [RPTOFC_MV].[ALT_ID_CD],
[RPTOFC_MV].[ROLTR], [RPTOFC_MV].[RPT_TO_OCD], [RPTOFC_MV].[RPTD_HLVL_GRPG_CD]

FROM [DSPN_DM], [DAYRANGE_DM], [RPTOFC_MV], [MTHLFOPT_SUM]

Where MTHLFOPT_SUM.FO_DAY_RANGE_SYS_NUM = DAYRANGE_DM.DAY_RANGE_SYS_NUM and
MTHLFOPT_SUM.DSPN_SYS_NUM = DSPN_DM.DSPN_SYS_NUM and MTHLFOPT_SUM.OFC_SYS_NUM =
RPTOFC_MV.OFC_SYS_NUM
```

For best performance, I took this select statement and integrated it into an update statement to save the output to the datacontent table. This improves the performance of the prototype web front-end and python migration script that I discuss in a later section of this paper. Here is the same query wrapped up into an update statement:

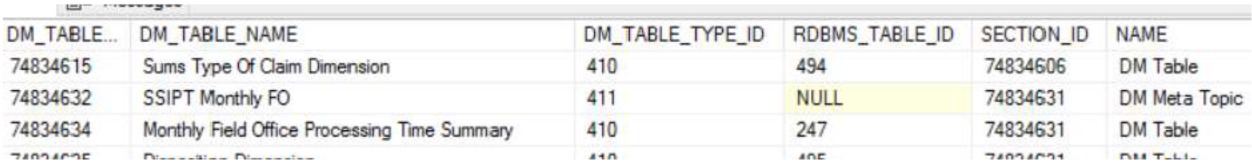


```
--update query text for data model sections
--this updates just one data model section 77748808
 -- remove "and dc.sectionid = 77748808" to update for all
 Update dc
 set [QueryText] = x.Query
 from dbo.DocumentContent dc
 inner join
 (
 select s.[Section_ID],
'Query' = case when DisplayFields is null then ' no data' --tables joins filters
            when not Filters is null and not Joins is null then
            concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins], ' and ', [Filters])
            when not Filters is null and Joins is null and SortFields is null then
            concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Filters])
            when Filters is null and not Joins is null then
            concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins])
            when Filters is null and Joins is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables])
            Else  concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins], ' and ', [Filters]) End
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [dbo].[DocumentContent] c on s.SECTION_ID = c.SectionID
where s.SECTION_TYPE_ID = 4
 ) as x on dc.SectionID = x.SECTION_ID and  dc.sectionid = 77748808
```

You will note that I put square brackets around the table and columns names. This is just a preventative measure. Most database systems do not support spaces in table and column names and if there are spaces, you have to put brackets around the names for the queries to work.

Data Model Section Types with Meta Topic Tables

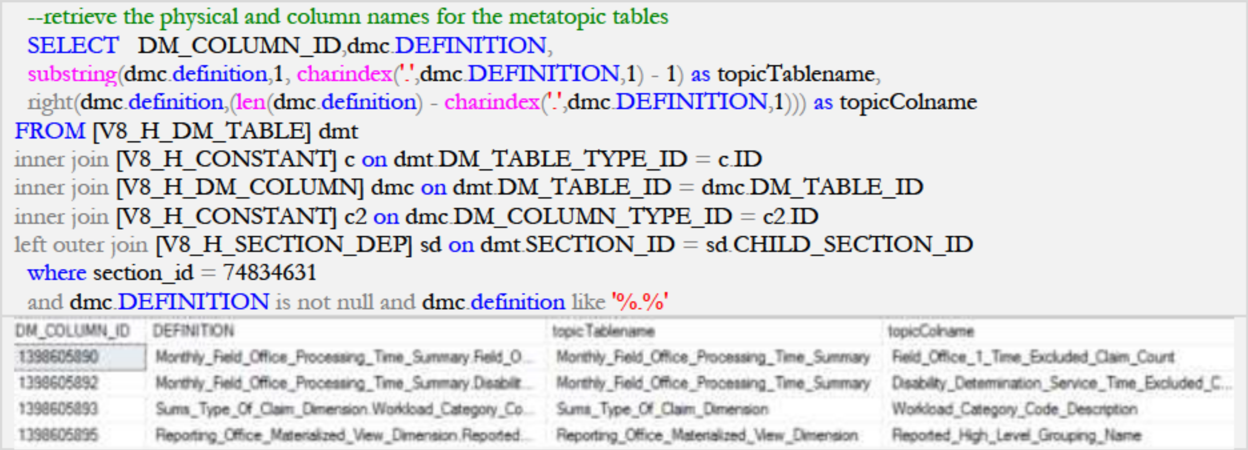
One thing I noticed about the Data models is that it categorized the data source tables into two different types either as a 'DM Table' or 'DM Meta Topic' table. The query we just walked through is sufficient for the tables categorized as a 'DM Table'. The tables categorized as Meta Topic are not physical tables but a view of one or more data models. These are harder to trace down to the physical data source. If you look at the output below, section 74834631 has two tables listed, one of which is **SSIPT MONTHLY FO,** which is a Meta Topic table and **Monthly Field Office Processing Time Summary,** which is a DM Table. We can trace the DM table to its physical source, which is evident as it has a foreign-key entry in the RDBMS_TABLE_ID column.



DM_TABLE...	DM_TABLE_NAME	DM_TABLE_TYPE_ID	RDBMS_TABLE_ID	SECTION_ID	NAME
74834615	Sums Type Of Claim Dimension	410	494	74834606	DM Table
74834632	SSIPT Monthly FO	411	NULL	74834631	DM Meta Topic
74834634	Monthly Field Office Processing Time Summary	410	247	74834631	DM Table
74834635	Disposition Dimension	410	495	74834631	DM Table

For the Metatopic tables, it may be useful to identify the tables and columns even though they contain spaces and special characters because depending on your target environment, you may want to recreate those views. For instance, you could use this to map to clusters in a WebFocus target solution. If you do stage those views with the same names, the queries would work. Even if you use different names, you could modify the scripts to replace the old name with the new name. In this instance, you could continue to use the same query for the tables in both categories.

However, if you need to identify the physical table and column names from the source table and columns used in the Meta Topic table then you can either leverage its parent data model section, or alternatively get creative with another query. I am going to show one way to do this with a query. There is always more than one way in the SQL world to get something done. Each column associated with a Meta Topic table has a **Definition** field that contains the physical table name and column name separated by period. This field is only populated for the Meta Topic tables and is null in the rows that are categorized as DM Tables. Use the following query to separate the tables and columns into separate fields.



```
  --retrieve the physical and column names for the metatopic tables
  SELECT   DM_COLUMN_ID,dmc.DEFINITION,
  substring(dmc.definition,1, charindex('.',dmc.DEFINITION,1) - 1) as topicTablename,
  right(dmc.definition,(len(dmc.definition) - charindex('.',dmc.DEFINITION,1))) as topicColname
FROM [V8_H_DM_TABLE] dmt
inner join [V8_H_CONSTANT] c on dmt.DM_TABLE_TYPE_ID = c.ID
inner join [V8_H_DM_COLUMN] dmc on dmt.DM_TABLE_ID = dmc.DM_TABLE_ID
inner join [V8_H_CONSTANT] c2 on dmc.DM_COLUMN_TYPE_ID = c2.ID
left outer join [V8_H_SECTION_DEP] sd on dmt.SECTION_ID = sd.CHILD_SECTION_ID
  where section_id = 74834631
  and dmc.DEFINITION is not null and dmc.definition like '%.%'
```

DM_COLUMN_ID	DEFINITION	topicTablename	topicColname
1398605890	Monthly_Field_Office_Processing_Time_Summary.Field_O...	Monthly_Field_Office_Processing_Time_Summary	Field_Office_1_Time_Excluded_Claim_Count
1398605892	Monthly_Field_Office_Processing_Time_Summary.Disabilit...	Monthly_Field_Office_Processing_Time_Summary	Disability_Determination_Service_Time_Excluded_C...
1398605893	Sums_Type_Of_Claim_Dimension.Workload_Category_Co...	Sums_Type_Of_Claim_Dimension	Workload_Category_Code_Description
1398605895	Reporting_Office_Materialized_View_Dimension.Reported...	Reporting_Office_Materialized_View_Dimension	Reported_High_Level_Grouping_Name

Then you could modify to combine with the earlier query for the "DM TABLE' as a derived table and use an outer join to get the display fields and table names if needed. My sample report does not need this so I will continue on to the next section type.

Query Section Type

This may be a little tricky to describe as I am going to talk about writing a SQL Query to recreate a query in EPM. I will distinguish between the two by calling the query I write as a **SQL Query** and the query that I am trying to recreate as just a plain '**query**' even though in the end they are both SQL queries. You retrieve table and column information from query sections similar to the data model sections. Queries sections contain display fields, tables, table joins, sorts, and filters.

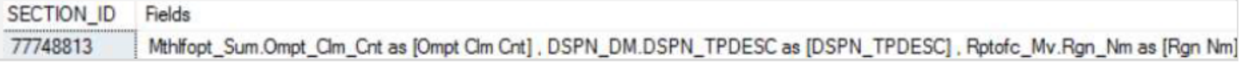
There are associative tables that connect the column fields to the table fields for each section. The V8_H_QRYCOL_DMCOL table maps the query columns to the data model columns. The V8_H_QRYCOL_RSCOL table maps the query columns to the results columns. To figure out the physical columns, you have to map the RDBMS_COLUMN_ID column in the V8_H_DM_COLUMN table to the RDBMS_COLUMN_ID column in the V8_H_DB_COLUMN table. The V8_H_DM_COLUMN column contains rows for the columns used by each data model section. The V8_H_DB_COLUMN column contains rows for each physical database. The best way is to illustrate this is with a SQL query. Here are the SQL queries that you can use to retrieve and format each query part. By format, I mean that we are grouping all the content into one large string value for each query section. When this value is output, you can be paste into a SQL query tool and use like any other SQL query.

For the SQL queries, we will return to using the sample document with a document_id of 1409716. This document has one query section type and its section_id is 77748813.

Here is the SQL query to retrieve the display fields and then rollup those fields to one row to format for each query section.

```sql
--retrieve display fields for query section type
select t1.SECTION_ID, string_agg(t1.field,', ') as Fields
from
(
select distinct s.SECTION_ID, sd.PARENT_SECTION_ID, sd.PARENT_SECTION_NAME, sd.PARENT_TYPE_ID,
 cast(concat(replace(replace(replace(replace(qc.SQL_DEFINITION,'        ',' '),'    ',' '),'  ',' '),'  ',' '), ' as
[',qc.QRY_COLUMN_NAME, '] ') as varchar(max)) as field
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_QRY_QUERY] q on q.SECTION_ID = s.SECTION_ID
inner join [V8_H_QRY_COLUMN] qc on q.QRY_QUERY_ID = qc.QRY_QUERY_ID
left outer join V8_H_QRYCOL_DMCOL qdm on qc.QRY_COLUMN_ID = qdm.QRY_COLUMN_ID
left outer join [V8_H_SECTION_DEP] sd on s.SECTION_ID = sd.CHILD_SECTION_ID
where SECTION_TYPE_ID = 8 and qc.DESCRIPTOR_FLAG in (102,103)
and s.SECTION_ID = 77748813) as t1
group by t1.SECTION_ID
```

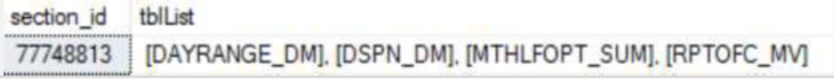
Results:



SECTION_ID	Fields
77748813	Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm Cnt] , DSPN_DM.DSPN_TPDESC as [DSPN_TPDESC] , Rptofc_Mv.Rgn_Nm as [Rgn Nm]

Query Section - Tables

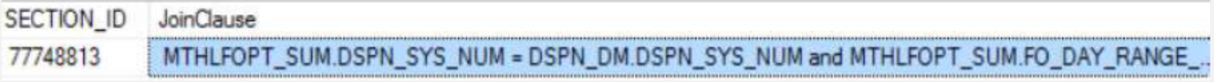
```sql
--format tables for database query
select z.section_id, cast(string_agg(concat('[',z.TableName,']'), ', ') as varchar(max)) as tblList
from
            (
select distinct s.section_Id,
dbt.RDBMS_TABLE_NAME as TableName
from [V8_H_SECTION] s
inner join [V8_H_SECTION_DEP] sd on s.SECTION_ID = sd.CHILD_SECTION_ID
inner join [V8_H_DM_JOIN] dmj on sd.PARENT_SECTION_ID = dmj.SECTION_ID
inner join [V8_H_HPSU_COLUMN] hco on dmj.FROM_HPSU_COLUMN_ID = hco.HPSU_COLUMN_ID
inner join [V8_H_DM_COLUMN] dmc on hco.HPSU_COLUMN_ID = dmc.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc on dmc.RDBMS_COLUMN_ID = dbc.RDBMS_COLUMN_ID
inner join [V8_H_DB_TABLE] dbt on dbc.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
where s.SECTION_TYPE_ID = 8 and s.SECTION_ID = 77748813
union --union table to ensure we tables from parent section
select distinct s.section_Id,
dbt2.RDBMS_TABLE_NAME
from [V8_H_SECTION] s
inner join [V8_H_SECTION_DEP] sd on s.SECTION_ID = sd.CHILD_SECTION_ID
inner join [V8_H_DM_JOIN] dmj on sd.PARENT_SECTION_ID = dmj.SECTION_ID
inner join [V8_H_HPSU_COLUMN] hco on dmj.FROM_HPSU_COLUMN_ID = hco.HPSU_COLUMN_ID
inner join [V8_H_HPSU_COLUMN] hco2 on dmj.TO_HPSU_COLUMN_ID = hco2.HPSU_COLUMN_ID
inner join [V8_H_DM_COLUMN] dmc2 on hco2.HPSU_COLUMN_ID = dmc2.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc2 on dmc2.RDBMS_COLUMN_ID = dbc2.RDBMS_COLUMN_ID
inner join [V8_H_DB_TABLE] dbt2 on dbc2.RDBMS_TABLE_ID = dbt2.RDBMS_TABLE_ID
where s.SECTION_TYPE_ID = 8 and s.SECTION_ID = 77748813) as z group by z.SECTION_ID
```



section_id	tblList
77748813	[DAYRANGE_DM], [DSPN_DM], [MTHLFOPT_SUM], [RPTOFC_MV]

Query Section - Joins

```sql
--query section - retrieve able joins and format for query
select z.SECTION_ID, cast(string_agg(z.joinCl, ' and ') as varchar(max)) as JoinClause from
(select distinct s.section_Id,
cast(concat(dbt.RDBMS_TABLE_NAME, '.',dbc.RDBMS_COLUMN_NAME, ' =
',dbt2.RDBMS_TABLE_NAME,'.' ,dbc2.RDBMS_COLUMN_NAME) as varchar(max)) as joinCl
from [V8_H_SECTION] s
inner join [V8_H_SECTION_DEP] sd on s.SECTION_ID = sd.CHILD_SECTION_ID
inner join [V8_H_DM_JOIN] dmj on sd.PARENT_SECTION_ID = dmj.SECTION_ID
inner join [V8_H_HPSU_COLUMN] hco on dmj.FROM_HPSU_COLUMN_ID = hco.HPSU_COLUMN_ID
inner join [V8_H_HPSU_COLUMN] hco2 on dmj.TO_HPSU_COLUMN_ID = hco2.HPSU_COLUMN_ID
inner join [V8_H_DM_COLUMN] dmc on hco.HPSU_COLUMN_ID = dmc.DM_COLUMN_ID
inner join [V8_H_DM_COLUMN] dmc2 on hco2.HPSU_COLUMN_ID = dmc2.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc on dmc.RDBMS_COLUMN_ID = dbc.RDBMS_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc2 on dmc2.RDBMS_COLUMN_ID = dbc2.RDBMS_COLUMN_ID
inner join [V8_H_DB_TABLE] dbt on dbc.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
inner join [V8_H_DB_TABLE] dbt2 on dbc2.RDBMS_TABLE_ID = dbt2.RDBMS_TABLE_ID
where s.SECTION_TYPE_ID = 8 and s.SECTION_ID = 77748813) as z
group by z.section_id
```

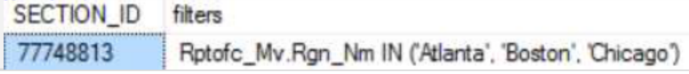


SECTION_ID	JoinClause
77748813	MTHLFOPT_SUM.DSPN_SYS_NUM = DSPN_DM.DSPN_SYS_NUM and MTHLFOPT_SUM.FO_DAY_RANGE_...

Text: MTHLFOPT_SUM.DSPN_SYS_NUM = DSPN_DM.DSPN_SYS_NUM and MTHLFOPT_SUM.FO_DAY_RANGE_SYS_NUM = DAYRANGE_DM.DAY_RANGE_SYS_NUM and MTHLFOPT_SUM.OFC_SYS_NUM = RPTOFC_MV.OFC_SYS_NUM

Query Section - Filters

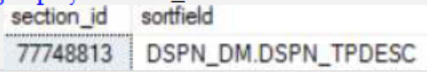
```
--query section  get filters and format
select x.SECTION_ID, string_agg(cast(JoinClause as varchar(max)),' and ') as filters
from
( select s.SECTION_ID,  replace(replace(replace(replace(qc.SQL_DEFINITION,'       ',''),'    ',''),' ',''),' ','') as JoinClause
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_QRY_QUERY] q on q.SECTION_ID = s.SECTION_ID
inner join [V8_H_QRY_COLUMN] qc on q.QRY_QUERY_ID = qc.QRY_QUERY_ID
where s.SECTION_TYPE_ID = 8 and DESCRIPTOR_FLAG = 101 and s.SECTION_ID = 77748813) AS x
group by x.section_id
```



SECTION_ID	filters
77748813	Rptofc_Mv.Rgn_Nm IN ('Atlanta', 'Boston', 'Chicago')

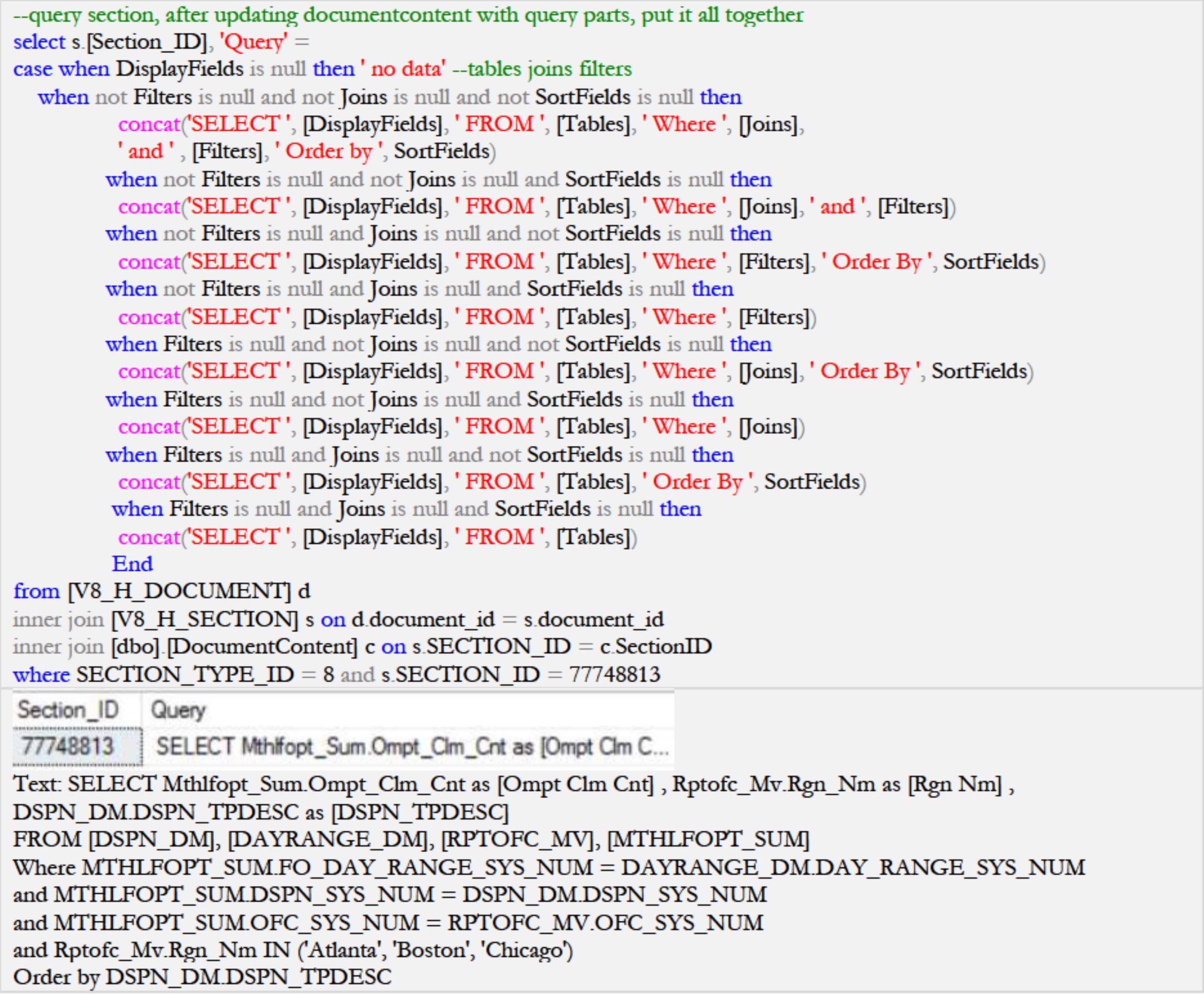
Query Section - Sort Columns

```
--query section get sort columns and format
select z.section_id, cast(string_agg(z.SQL_DEFINITION, ', ') as varchar(max)) as sortfield
from
(select distinct s.section_id, dbc.RDBMS_COLUMN_NAME, qc.SQL_DEFINITION
from [V8_H_SECTION] s
inner join [V8_H_QRY_QUERY] q on s.section_id = q.section_id
inner join [V8_H_QRY_COLUMN] qc on q.QRY_QUERY_ID = qc.QRY_QUERY_ID
inner join [V8_H_QRYCOL_DMCOL] qcdm on qc.QRY_COLUMN_ID = qcdm.QRY_COLUMN_ID
inner join [V8_H_DM_COLUMN] dmc on qcdm.HPSU_COLUMN_ID = dmc.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc on dmc.RDBMS_COLUMN_ID = dbc.RDBMS_COLUMN_ID
inner join [V8_H_DB_TABLE] dbt on dbc.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
inner join [V8_H_CONSTANT] hc on qc.DESCRIPTOR_FLAG = hc.ID
where s.SECTION_TYPE_ID = 8 and qc.DESCRIPTOR_FLAG = 103 and s.SECTION_ID = 77748813) as z
group by z.section_id
```



section_id	sortfield
77748813	DSPN_DM.DSPN_TPDESC

Now let us put all of the query parts together to recreate the query:



```
--query section, after updating documentcontent with query parts, put it all together
select s.[Section_ID], 'Query' =
case when DisplayFields is null then ' no data' --tables joins filters
   when not Filters is null and not Joins is null and not SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins],
             ' and ' , [Filters], ' Order by ', SortFields)
            when not Filters is null and not Joins is null and SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins], ' and ', [Filters])
            when not Filters is null and Joins is null and not SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Filters], ' Order By ', SortFields)
            when not Filters is null and Joins is null and SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Filters])
            when Filters is null and not Joins is null and not SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins], ' Order By ', SortFields)
            when Filters is null and not Joins is null and SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins])
            when Filters is null and Joins is null and not SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Order By ', SortFields)
             when Filters is null and Joins is null and SortFields is null then
             concat('SELECT ', [DisplayFields], ' FROM ', [Tables])
             End
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [dbo].[DocumentContent] c on s.SECTION_ID = c.SectionID
where SECTION_TYPE_ID = 8 and s.SECTION_ID = 77748813
```

Section_ID	Query
77748813	SELECT Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm C...

```
Text: SELECT Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm Cnt] , Rptofc_Mv.Rgn_Nm as [Rgn Nm] ,
DSPN_DM.DSPN_TPDESC as [DSPN_TPDESC]
FROM [DSPN_DM], [DAYRANGE_DM], [RPTOFC_MV], [MTHLFOPT_SUM]
Where MTHLFOPT_SUM.FO_DAY_RANGE_SYS_NUM = DAYRANGE_DM.DAY_RANGE_SYS_NUM
and MTHLFOPT_SUM.DSPN_SYS_NUM = DSPN_DM.DSPN_SYS_NUM
and MTHLFOPT_SUM.OFC_SYS_NUM = RPTOFC_MV.OFC_SYS_NUM
and Rptofc_Mv.Rgn_Nm IN ('Atlanta', 'Boston', 'Chicago')
Order by DSPN_DM.DSPN_TPDESC
```

Now that we have successfully recreated our query, let us move on to the Results Section Type.

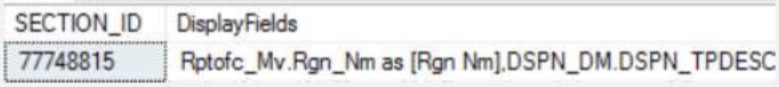
Results Section Type

The Results section types do not directly connect to data but is rather a way to format and view data from queries and data models. It also can include formulas to create custom calculations. Because users perform these calculations within the EPM client tool and not within SQL, we cannot easily port these formulas to the target system. Many users have expressed the need to identify these computed columns so I at a minimum, focused on identifying the display fields, filter columns, sort columns, and computed columns. Additionally, the SQL_Definition field contains too many variations to parse out the tables and columns, as I was able to with the metatopic tables for the data model. You could take it further if you wish and merge the data with the parent section to create a full query.

For this sample code, I will again work with Document_ID 1409716. The section_id is 77748815.

Result Section – Display Fields

```
--results section, get display fields
select s.SECTION_ID,  string_agg( cast(concat(replace(replace(replace(replace(qc.SQL_DEFINITION,'        ',' '),'    ',' '),'
',' '),' ',' '),' as [', rc.RS_COLUMN_NAME,']') as varchar(max)), ',') as DisplayFields
from [V8_H_DOCUMENT] d inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_RS_COLUMN] rc on rc.SECTION_ID = s.SECTION_ID
inner join [V8_H_QRYCOL_RSCOL] qrc on rc.RS_COLUMN_ID = qrc.RS_COLUMN_ID
inner join [V8_H_QRY_COLUMN] qc on qrc.QRY_COLUMN_ID = qc.QRY_COLUMN_ID
where s.SECTION_TYPE_ID = 10  and rc.DESCRIPTOR_FLAG in (102, 103)
and s.section_id = 77748815
```



SECTION_ID	DisplayFields
77748815	Rptofc_Mv.Rgn_Nm as [Rgn Nm],DSPN_DM.DSPN_TPDESC

```
group by s.SECTION_ID
Text: Rptofc_Mv.Rgn_Nm as [Rgn Nm],DSPN_DM.DSPN_TPDESC as
[DSPN_TPDESC],Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm Cnt]
```

Result Section – Computed Fields

```
--results section, get computed fields
select s.section_id, string_agg(cast( concat(ci.FORMULA, ' as [', rc.RS_COLUMN_NAME, ']') as varchar(max)),', ') as
Computed
from [V8_H_DOCUMENT] d inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_RS_COLUMN] rc on rc.SECTION_ID = s.SECTION_ID
inner join [V8_H_RS_COMP_ITEM] ci on rc.RS_COLUMN_ID = ci.[RS_COLUMN_ID]
where s.SECTION_TYPE_ID = 10  and s.section_id = 77748815
group by s.SECTION_ID
--No computed fields for this section
```

Result Section – Sort Fields

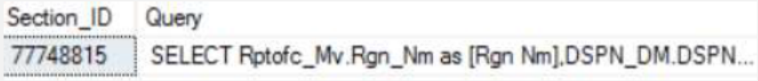
```
--results section, get sort fields
select s.section_id, string_agg(cast(concat(replace(replace(replace(replace(qc.SQL_DEFINITION,'        ',' '),'    ',' '),' ',' '),' ','
'),' ') as varchar(max)), ', ') as SortFields
from [V8_H_DOCUMENT] d inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_RS_COLUMN] rc on rc.SECTION_ID = s.SECTION_ID
inner join [V8_H_QRYCOL_RSCOL] qrc on rc.RS_COLUMN_ID = qrc.RS_COLUMN_ID
inner join [V8_H_QRY_COLUMN] qc on qrc.QRY_COLUMN_ID = qc.QRY_COLUMN_ID
where s.SECTION_TYPE_ID = 10 and rc.DESCRIPTOR_FLAG = 103 and s.section_id = 77748815
group by s.section_id
--No sort fields for this section
```

Result Section - Filters

```
--results section, get filters
select s.SECTION_ID, string_agg(cast(concat(rc.RS_COLUMN_NAME, ' in (', w.LIMIT_VALUES, ')') as varchar(max)), '
and ') as Filters
from [V8_H_DOCUMENT] d inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_RS_COLUMN] rc on rc.SECTION_ID = s.SECTION_ID
inner join [V8_H_RS_LIMIT_COL] w on rc.RS_COLUMN_ID = w.RS_COLUMN_ID
where s.SECTION_TYPE_ID = 10  and s.section_id = 77748815
group by s.SECTION_ID
--No filters for this section
```

Result Section – Combine Display Fields, Computed Fields, and Sort Fields

```
--merge result parts
select s.[Section_ID], 'Query' =
case when DisplayFields is null then ' no data'
when not Filters is null and not ComputedFields is null and not SortFields is null
then concat('SELECT ', [DisplayFields], ',', ComputedFields, ' Where ', Filters, ' Order By ', SortFields )
when not Filters is null and ComputedFields is null and SortFields is null
then concat('SELECT ', [DisplayFields], ' Where ', Filters)
when not Filters is null and not ComputedFields is null and SortFields is null
then concat('SELECT ', [DisplayFields], ',', ComputedFields, ' Where ', Filters)
when not Filters is null and ComputedFields is null and not SortFields is null
then concat('SELECT ', [DisplayFields], ' Where ', Filters, ', Order By ', SortFields )
when Filters is null and not ComputedFields is null and not SortFields is null
then concat('SELECT ', [DisplayFields], ',', ComputedFields,  ' Order By ', SortFields )
when Filters is null and ComputedFields is null and SortFields is null
then concat('SELECT ', [DisplayFields])
when Filters is null and not ComputedFields is null and SortFields is null
then concat('SELECT ', [DisplayFields], ',', ComputedFields)
when Filters is null and ComputedFields is null and not SortFields is null
then concat('SELECT ', [DisplayFields], ' Order By ', SortFields )
Else  concat('SELECT ', [DisplayFields], ' FROM ', [Tables], ' Where ', [Joins], ' and ', [Filters]) End
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [dbo].[DocumentContent] c on s.SECTION_ID = c.SectionID and c.SectionID = 77748815
where s.section_type_id = 10
```



Section_ID	Query
77748815	SELECT Rptofc_Mv.Rgn_Nm as [Rgn Nm],DSPN_DM.DSPN...

```
Text: SELECT Rptofc_Mv.Rgn_Nm as [Rgn Nm],DSPN_DM.DSPN_TPDESC as
[DSPN_TPDESC],Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm Cnt]
```

Here is an example of output for a result section type that contains a computed column:

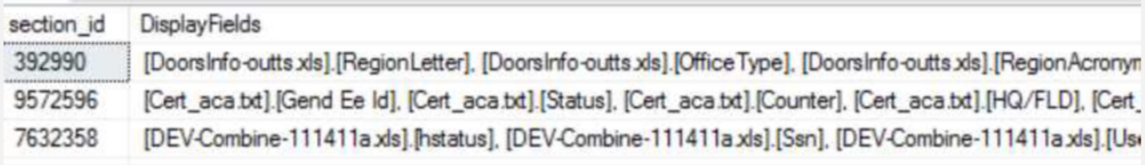
```
SELECT V8_Prop_Value.Value0 as [Email],V8_Subject.Name as [Name],if (Email!="EMPTY") {Email} else {""}
as [User Email]
```

This wraps up the results section. There are couple of section types that I will touch on briefly.

Import Section Type

Import section types contain information about data imported from other sources such as a spreadsheet. At a minimum, we can use a query against the EPM database to provide the file name and fields used from that file. I have provided an example query below where I am returning the first 100 sections that are imports and retrieving the fields and file that are used and then rolling up those fields up into a single line for each section.

```
select top 100 x1.section_id, string_agg(cast( concat('[',x1.HPSU_TABLE_NAME, '].[',x1.HPSU_COLUMN_NAME,']') as
varchar(max)), ', ') as DisplayFields
 from (
select distinct ht.DOCUMENT_ID, ht.HPSU_TABLE_ID,
ht.HPSU_TABLE_NAME, ht.HPSU_TABLE_TYPE_ID, ct.NAME as tableType,
hc.HPSU_COLUMN_ID, hc.HPSU_COLUMN_NAME, s.SECTION_ID, s.SECTION_NAME
from
 [V8_H_HPSU_TABLE] ht
inner join [V8_H_CONSTANT] ct on ht.HPSU_TABLE_TYPE_ID = ct.ID
inner join [V8_H_HPSU_COLUMN] hc on ht.HPSU_TABLE_ID = hc.HPSU_TABLE_ID
inner join [V8_H_SECTION] s on ht.DOCUMENT_ID = s.DOCUMENT_ID and s.SECTION_NAME =
ht.HPSU_TABLE_NAME
and s.SECTION_TYPE_ID = ht.HPSU_TABLE_TYPE_ID
where ht.HPSU_TABLE_TYPE_ID = 5
 ) as x1 group by x1.SECTION_ID
```



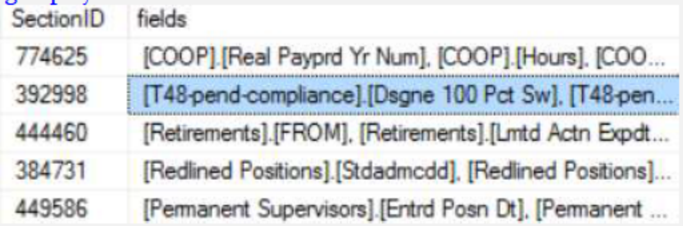
section_id	DisplayFields
392990	[DoorsInfo-outts.xls].[RegionLetter], [DoorsInfo-outts.xls].[OfficeType], [DoorsInfo-outts.xls].[RegionAcronym
9572596	[Cert_aca.txt].[Gend Ee Id], [Cert_aca.txt].[Status], [Cert_aca.txt].[Counter], [Cert_aca.txt].[HQ/FLD], [Cert_
7632358	[DEV-Combine-111411a.xls].[hstatus], [DEV-Combine-111411a.xls].[Ssn], [DEV-Combine-111411a.xls].[Us

```
--Text from the first result: [DoorsInfo-outts.xls].[RegionLetter], [DoorsInfo-outts.xls].[OfficeType], [DoorsInfo-
outts.xls].[RegionAcronym], [DoorsInfo-outts.xls].[RegionNumber], [DoorsInfo-outts.xls].[OfficeTypeCode], [DoorsInfo-
outts.xls].[AreaNumber], [DoorsInfo-outts.xls].[OfficeCode], [DoorsInfo-outts.xls].[ReportsToOfficeCode], [DoorsInfo-
outts.xls].[OfficeName]
```

Table Section Type

The last section type I am going to discuss is the Table Section Type. These section types are locally created tables. At a minimum, we can identify the fields and local table name. Here is a query the returns the first 100 sections that are table types.

```
--table type
select top 100 x.SectionID, STRING_AGG(x.field, ', ') as fields
from(
 select distinct dc.sectionid, hc.HPSU_COLUMN_ID,
 cast(concat('[',ht.HPSU_TABLE_NAME,']','.','[',hc.HPSU_COLUMN_NAME,']') as varchar(max)) as field
from [dbo].[DocumentContent] dc
  inner join [V8_H_SECTION] s on dc.sectionid = s.section_id
  inner join [V8_H_CONSTANT] c on s.SECTION_TYPE_ID = c.ID
  inner join [V8_H_HPSU_TABLE] ht on s.SECTION_ID = ht.HPSU_TABLE_ID
  inner join [V8_H_HPSU_COLUMN] hc on ht.HPSU_TABLE_ID = hc.HPSU_Table_ID
  where s.SECTION_type_id = 11
 ) as x
 group by x.SectionID
```



SectionID	fields
774625	[COOP].[Real Payprd Yr Num], [COOP].[Hours], [COO...
392998	[T48-pend-compliance].[Dsgne 100 Pct Sw], [T48-pen...
444460	[Retirements].[FROM], [Retirements].[Lmtd Actn Expdt...
384731	[Redlined Positions].[Stdadmcdd], [Redlined Positions]...
449586	[Permanent Supervisors].[Entrd Posn Dt], [Permanent ...

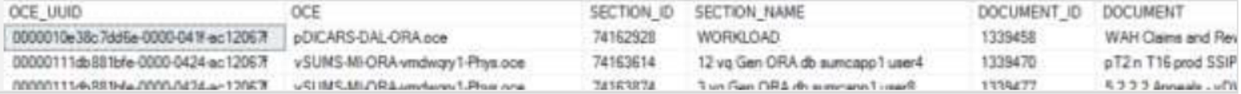
Other Section Types.

There are other section types that I was not able to recreate and that includes Pivots, Charts, Report, and Dashboards. Recreating these sections requires access to the report files themselves, which is not something I had.

Identifying DataSources

The last piece to recreating queries is the datasource they leverage. In EPM, the datasource connections to the databases are called open catalog extensions (OCEs). Here is the query to identify the OCE for each query section:

```
--identifying datasource
select top 100 oce.OCE_UUID, co.NAME as OCE,s.SECTION_ID, s.SECTION_NAME,d.DOCUMENT_ID,
d.DOCUMENT
from [V8_QRY_DB_CONN] oce
inner join [V8_CONT_VERSION] c on oce.CONTAINER_UUID = c.CONTAINER_UUID
and oce.VERSION_NUMBER = c.VERSION_NUMBER
inner join [V8_H_DOCUMENT] d on c.CONTAINER_UUID = d.UUID and c.VERSION_NUMBER =
d.[VERSION]
inner join [V8_CONTAINER] co on oce.OCE_UUID = co.CONTAINER_UUID
inner join [V8_H_SECTION] s on d.DOCUMENT_ID = s.DOCUMENT_ID
where not oce.OCE_UUID is null and oce.[QUERY_NAME] = s.SECTION_NAME
```



OCE_UUID	OCE	SECTION_ID	SECTION_NAME	DOCUMENT_ID	DOCUMENT
0000010e38c7dd6e-0000-041f-ac12067f	pDICARS-DAL-ORA.oce	74162928	WORKLOAD	1339458	WAH Claims and Rev
00000111db881bfe-0000-0424-ac12067f	vSUMS-MI-ORA-vmdwqry1-Phys.oce	74163614	12 vq Gen ORA db sumcapp1 user4	1339470	pT2n T16 prod SSIP
00000111db881bfe-0000-0424-ac12067f	vSUMS-MI-ORA-vmdwqry1-Phys.oce	74163874	3 vq Gen ORA db sumcapp1 user8	1339477	5.2.2.2 Appeals - vOW

Restful Web Services

As I mentioned in the introduction, enterprise report solutions use restful web APIs to ease administrative tasks. One of the target migration environments here at SSA is WebFocus. I will show how one can access WebFocus using Restful APIs to migrate query section types from EPM.

This process will be similar with any Enterprise Reporting Tool you use. You will need to dig into their administration and development documentation and adapt these examples for the desired target environment. Here are some of the links to documentation on using web services APIs for some of the popular vendors:

- IBI WebFocus https://infocenter.informationbuilders.com/wf80/index.jsp?topic=%2Fpubdocs%2FREST_WebServices%2Fsource%2Fopener.htm
- Tableau https://onlinehelp.tableau.com/current/api/rest_api/en-us/REST/rest_api_concepts_publish.htm
- Microsoft SQL Server Reporting Services https://docs.microsoft.com/en-us/sql/reporting-services/developer/rest-api?view=sql-server-2017

Converting Query Section Type to WebFocus

The first step is to take a query from EPM and convert it into WebFocus syntax. You also have to map the datasource as well. I have already staged all the queries for all query section types into QueryText field of the DocumentContent table. I have another field in the same table called WebFocusText that I will use to save the results of the converted text.

```
update [dbo].[DocumentContent]
set [WebFocusText] = x.webFocusText
from [dbo].[DocumentContent] dc1 inner join
(
 select dc.SectionID, dc.SectionName, dc.DataSourceID, dc.QueryText, dc.DisplayFields,
 ds.HostName, ds.OceName,
case when  dc.DisplayFields is null then 'NO DATA'
else concat('-DEFAULT &CLIENTID = 183;
- ENGINE SQLORA SET DEFAULT_CONNECTION DWEWAND
SQL SQLORA PREPARE SQLOUT FOR
', dc.[QueryText], '
END

TABLE FILE SQLOUT
PRINT ', replace([DisplayFields], ',',' ') , '
END')
```

```
 end as webFocusText
 FROM [dbo].[DocumentContent] dc
 inner join [V8_H_SECTION] s on dc.SectionID = s.SECTION_ID
 inner join [dbo].[DataSource] ds on dc.DataSourceID = ds.DatasourceID) as x
 on x.SectionID = dc1.SectionID
where dc1.sectionid = 77748813
```

```
--results of webfocus conversion
 select sectionid, webFocusText
from [dbo].[DocumentContent] where sectionid = 77748813
Output Text:
-DEFAULT &CLIENTID = 183;
- ENGINE SQLORA SET DEFAULT_CONNECTION DWEWAND
SQL SQLORA PREPARE SQLOUT FOR  SELECT Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm Cnt] ,
Rptofc_Mv.Rgn_Nm as [Rgn Nm] , DSPN_DM.DSPN_TPDESC as [DSPN_TPDESC]  FROM [DSPN_DM],
[DAYRANGE_DM], [RPTOFC_MV], [MTHLFOPT_SUM] Where MTHLFOPT_SUM.FO_DAY_RANGE_SYS_NUM
= DAYRANGE_DM.DAY_RANGE_SYS_NUM and MTHLFOPT_SUM.DSPN_SYS_NUM =
DSPN_DM.DSPN_SYS_NUM and MTHLFOPT_SUM.OFC_SYS_NUM = RPTOFC_MV.OFC_SYS_NUM and
Rptofc_Mv.Rgn_Nm IN ('Atlanta', 'Boston', 'Chicago') Order by DSPN_DM.DSPN_TPDESC   END

TABLE FILE SQLOUT  PRINT
Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm Cnt]   Rptofc_Mv.Rgn_Nm as [Rgn Nm]   DSPN_DM.DSPN_TPDESC as
[DSPN_TPDESC]    END
```

Now we can use a Python query. This Python query stores a folder value that it will use to connect to a SQL server, find all the documents in that folder and then loop through all the documents. As it loops through all the documents, it will find all the sections and loop through each section. As it loops through each section, it will pull the WebFocus content and use it to create a virtual WebFocus fex file in memory, connect to the WebFocus server and upload the fex file to the desired virtual folder on the WebFocus server. The complete python script is available in Appendix C. Let us look at portions of the script to discuss the critical aspects.

This portion defines the libraries needed within Python to interact with SQL Server, the file system, and WebFocus. I used the minidom library to parse through the webpage response and retrieve the security token. This security token is required after the first connection to enable the actual upload activity.

```
#import needed libraries
import pypyodbc
import csv
import sys
import pandas as pd
import requests

#needed to encode fex file for WebFocus. Encoding is not needed for Tableau
import base64

from requests.auth import AuthBase
from requests.auth import HTTPBasicAuth
from requests.auth import HTTPDigestAuth

#needed to parse through web page response to retrieve security token
import xml.etree.ElementTree as ET
from xml.dom import minidom
```

This code specifies information needed to connect to WebFocus, which includes the user credentials and URL. You need to replace the username and password with the credentials you are using. Then we take the username and URL and build the connection string that the script will send to the WebFocus server to make a web service request.

```
##define the api endpoint and user credentials

devuser = 'username'
devpw = 'password'
strbaseurl = 'http://<webserver_url>/ibi_apps/rs/ibfs'
wfDevURL = strbaseurl + '?IBIRS_action=signOn&IBIRS_userName=' + devuser + "&IBIRS_password=" + devpw
```

Here we specify the folder in EPM that contains the documents we want to migrate and use that to build our SQL query to retrieve the SQL content.

```
#define folder to be migrated
mgfolder =  '/BI Central Office Folders/DCBFM/DCBFM FPA/Development/migrationScript_test'

getdocsSQL = "select document_id from [V8_H_DOCUMENT] where path = '" + mgfolder + "'"
```

Here we define our connection to SQL Server, make the call to connect to SQL Server, and then make a call to signonWF() which calls the function that does all of the migration work.

```
##connection string to migration database
conn = pypyodbc.connect("DRIVER={SQL Server};"
                    "SERVER=<database server name>;"
                    "DATABASE=<database name>;"
                    "Trusted_Connection=yes;")

cursor = conn.cursor()

#call function to get documents and perform migration
signonWF()
```

The code nested under the def sigonWF() function is the block of code that defines the signonWF(). It contains the logic that when called, performs most of the migration work. I will highlight critical pieces of code under this function. As a reminder, the entire script is in Appendix C.

This code makes the initial connection to the Webfocus server and checks for a response. If there is a successful response, it saves the output to a virtual text and xml file.

```
#defines the signonWF function
def signonWF():
    response = requests.post(wfDevURL)

    status = response.status_code

    if status == 200:

        print('successfully connected ', response.status_code)

        r = response.text
        #r.write('output.xml')
        cookie = response.cookies
        #print(cookie)
    root = ET.fromstring(r)

    tree = ET.ElementTree(root)
    #write to xml file so that minidom can parse it
    tree.write("output.xml")
```

This next part, extracts the session cookie and security token from the webserver response of the initial connection. Both the cookie and security token are needed in order to complete the upload. If either piece is missing, this script will not work.

```
    dom = minidom.parse("output.xml")
    xmltags = dom.getElementsByTagName('entry')
    stoken = [items.attributes['value'].value for items in xmltags if items.attributes['key'].value ==
"IBI_CSRF_Token_Value"]
    print('security token is ', stoken)
```

Next we take the dataset we received form executing our early SQL query and loop through all the documents to retrieve all of the query sections from the document.

```
    mdocs = pd.read_sql(getdocsSQL,con=conn)
    d = 0
    for row in mdocs.values:
        docid = row[0]
        #identify all query sections that have webfocus content for each document
        secsql = "SELECT concat(replace(d.DOCUMENT,' ','_'), '_sec_', replace(s.SECTION_NAME,' ','_')) as docTitle, \
        [WebFocusText] FROM [dbo].[DocumentContent] dc  inner join [V8_H_DOCUMENT] d on dc.DocumentID =
d.DOCUMENT_ID \
        inner join [V8_H_SECTION] s on dc.SectionID = s.SECTION_ID where not [WebFocusText] is null and not
WebFocusText = 'NO DATA' \
        and s.SECTION_TYPE_ID = 8 and dc.documentid = " + str(docid)

        wfData = pd.read_sql(secsql,con=conn)
```

As we look at each query section, we retrieve the WebFocus content and build the virtual fex file. We must then encode this fex to utf-8. If you do not encode this file, the upload process will insert an empty file. Not all webservers require this encoding so you should check the documentation of the product you are using.

```
        i = 0
        for row in wfData.values:

            #get report title
            description = row[0]
            filecontent = row[1]


            byteobj = base64.b64encode(filecontent.encode())
            fileobj = byteobj.decode("utf-8")
            #print(fileobj)

            reportpath = '/WFC/Repository/BIMigration/uploadtest/' + description + '.fex'
```

Next, we take the security token and fex file and create an XML object. Then we pass this object along with the cookie object within another request to the WebFocus server to upload the fex file. Then we move on to the next section.

```
            strtoken = stoken[0]

            secToken = '&IBIWF_SES_AUTH_TOKEN=' + strtoken
            putObj = '<rootObject _jt="IBFSMRObject" description="' + description + '" type="FexFile"> <content
_jt="IBFSByteContent" char_set="Cp1252">' + fileobj + '</content> </rootObject>'
            upURL = strbaseurl + reportpath + '?IBIRS_action=put&IBIRS_object=' + putObj + secToken
            #print(upURL)

            uploadresponse = requests.post(upURL, cookies=cookie)
            #file.close
            #print(uploadresponse.text)
            #increment for next section
            i+=1

        #increment for next document
        d+=1
```

When there are no more sections, the loop exits. The final step is to close the database connection and print a message indicating that the script is finished.

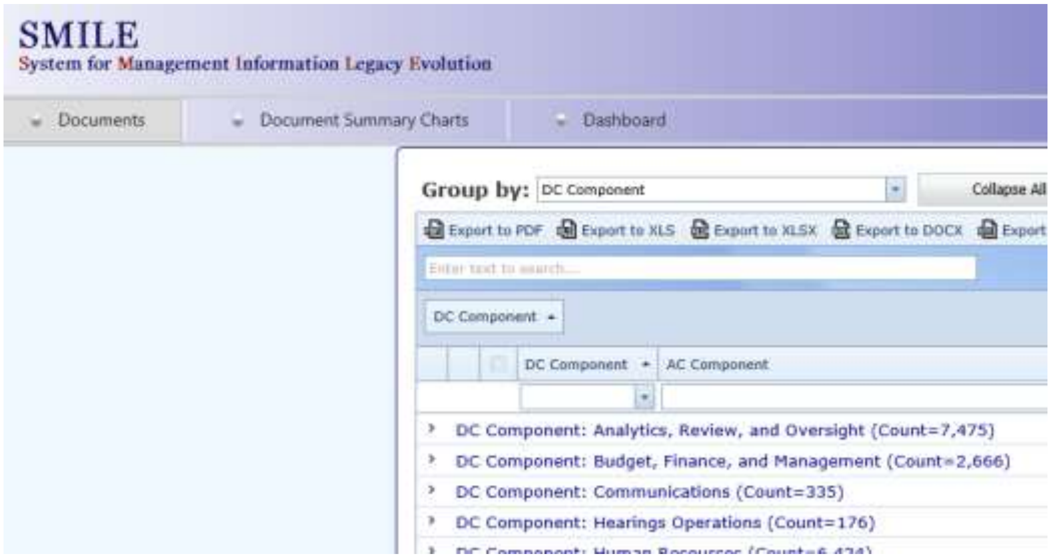
```
conn.commit()
cursor.close()
conn.close()

print("all done migrating for this folder, please check the webfocus server")
```

Now we can check the Webfocus server to see our files.

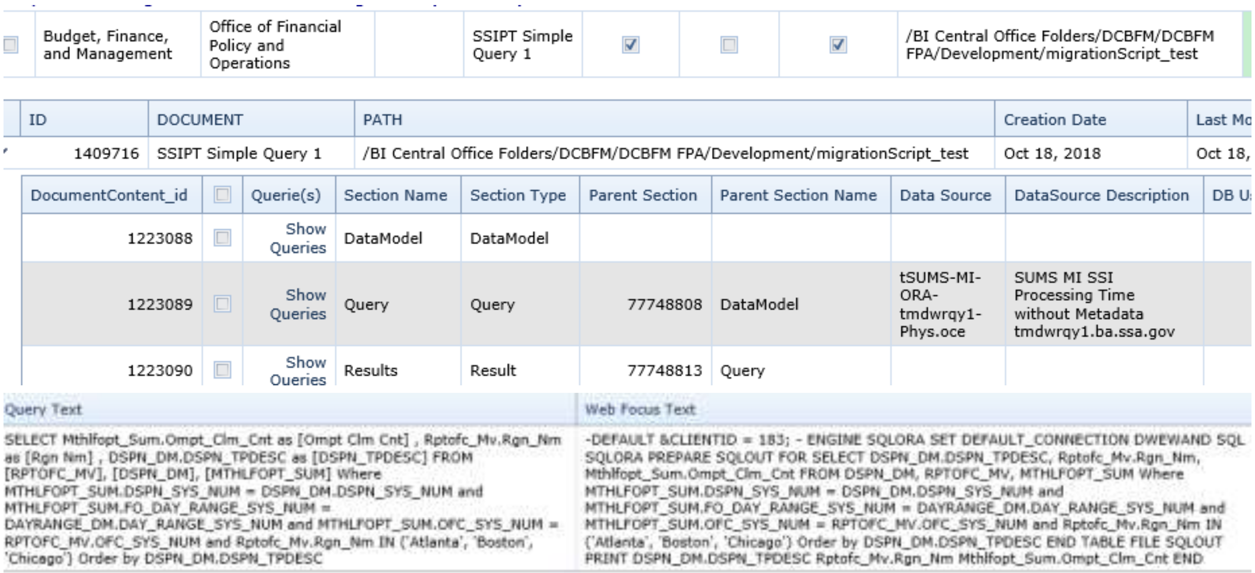
SMILE - Visualizing the migration metadata

The ADAL was fortunate to have the support of (b) (6), a web developer from the Office Disability Policy and Management Information (ODPMI) within the Office of Disability Policy (ODP). (b) (6) built a web front end to allow visualization of some of the content we collected about the EPM documents (b) (6) creatively named this application System for Management Information Legacy Evolution (SMILE). Within SSA's network, you can access this prototype at http://<smile_url>/Documents.aspx . Below is a screen shot:



- DC Component: Analytics, Review, and Oversight (Count=7,475)
- DC Component: Budget, Finance, and Management (Count=2,666)
- DC Component: Communications (Count=335)
- DC Component: Hearings Operations (Count=176)
- DC Component: Human Resources (Count=6,474)

(b) (6) also provided a way to drill down into each section and then view the recreated content:



□	Budget, Finance, and Management	Office of Financial Policy and Operations		SSIPT Simple Query 1	☑	□	☑	/BI Central Office Folders/DCBFM/DCBFM FPA/Development/migrationScript_test

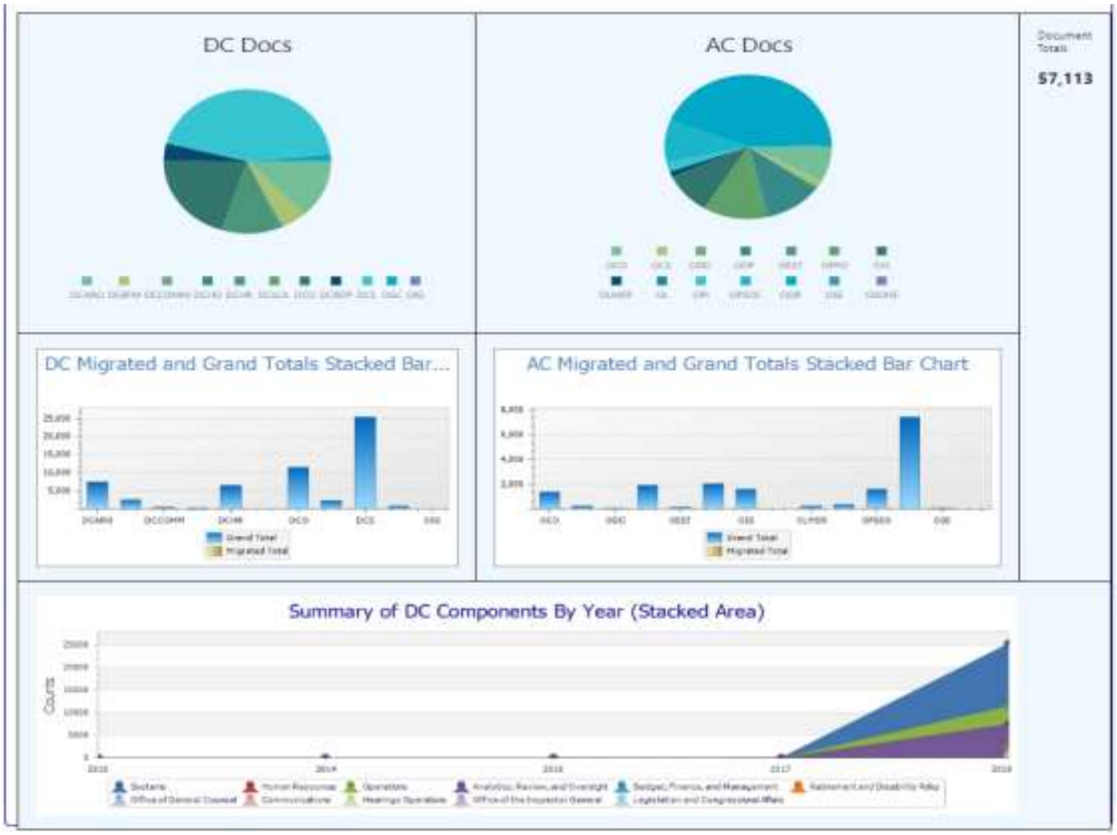
ID	DOCUMENT	PATH	Creation Date	Last Mo
1409716	SSIPT Simple Query 1	/BI Central Office Folders/DCBFM/DCBFM FPA/Development/migrationScript_test	Oct 18, 2018	Oct 18,

DocumentContent_id	□	Querie(s)	Section Name	Section Type	Parent Section	Parent Section Name	Data Source	DataSource Description	DB U:
1223088	□	Show Queries	DataModel	DataModel					
1223089	□	Show Queries	Query	Query	77748808	DataModel	tSUMS-MI-ORA-tmdwrqy1-Phys.oce	SUMS MI SSI Processing Time without Metadata tmdwrqy1.ba.ssa.gov	
1223090	□	Show Queries	Results	Result	77748813	Query			

Query Text	Web Focus Text
SELECT Mthlfopt_Sum.Ompt_Clm_Cnt as [Ompt Clm Cnt] , Rptofc_Mv.Rgn_Nm as [Rgn Nm] , DSPN_DM.DSPN_TPDESC as [DSPN_TPDESC] FROM [RPTOFC_MV], [DSPN_DM], [MTHLFOPT_SUM] Where MTHLFOPT_SUM.DSPN_SYS_NUM = DSPN_DM.DSPN_SYS_NUM and MTHLFOPT_SUM.FO_DAY_RANGE_SYS_NUM = DAYRANGE_DM.DAY_RANGE_SYS_NUM and MTHLFOPT_SUM.OFC_SYS_NUM = RPTOFC_MV.OFC_SYS_NUM and Rptofc_Mv.Rgn_Nm IN ('Atlanta', 'Boston', 'Chicago') Order by DSPN_DM.DSPN_TPDESC	-DEFAULT &CLIENTID = 183; - ENGINE SQLORA SET DEFAULT_CONNECTION DWEWAND SQL SQLORA PREPARE SQLOUT FOR SELECT DSPN_DM.DSPN_TPDESC, Rptofc_Mv.Rgn_Nm, Mthlfopt_Sum.Ompt_Clm_Cnt FROM DSPN_DM, RPTOFC_MV, MTHLFOPT_SUM Where MTHLFOPT_SUM.DSPN_SYS_NUM = DSPN_DM.DSPN_SYS_NUM and MTHLFOPT_SUM.FO_DAY_RANGE_SYS_NUM = DAYRANGE_DM.DAY_RANGE_SYS_NUM and MTHLFOPT_SUM.OFC_SYS_NUM = RPTOFC_MV.OFC_SYS_NUM and Rptofc_Mv.Rgn_Nm IN ('Atlanta', 'Boston', 'Chicago') Order by DSPN_DM.DSPN_TPDESC END TABLE FILE SQLOUT PRINT DSPN_DM.DSPN_TPDESC Rptofc_Mv.Rgn_Nm Mthlfopt_Sum.Ompt_Clm_Cnt END

This application is a C# MVC (model-view-controller) application written in Asp.Net and hosted on a windows web server within the Office of Systems in the SHE environment. Its data is stored on a SQL Server data hosted on a SQL Server 2017 server within the Office of Budget, Finance and Management.

This application implements the DevExpress grid and charts that consist of bar, pie, and area charts. There is also a dashboard that shows a summary of migrated reports.


DC Docs
AC Docs
Document Totals
57,113
DC Migrated and Grand Totals Stacked Bar...
25,000
20,000
15,000
10,000
5,000
DCARO
DCCOMM
DCHR
DCO
DCS
OIG
Grand Total
Migrated Total
AC Migrated and Grand Totals Stacked Bar Chart
8,000
6,000
4,000
2,000
OCD
OEST
OIS
OPSOS
Grand Total
Migrated Total
Summary of DC Components By Year (Stacked Area)
Counts
25000
20000
15000
10000
5000
0
2014
2016
2017
2018
Systems
Human Resources
Operations
Analytics, Review, and Oversight
Budget, Finance, and Management
Retirement and Disability Policy
Office of General Counsel
Communications
Hearings Operations
Office of the Inspector General
Legislation and Congressional Affairs

Resource Savings

Using some or all of these methods could result in saving time and staff resources that manually track the migration process and recreate the documents in another environment. For instance, consider if you used this process to move over 5,000 query sections. Let us estimate that it takes a user thirty minutes to identify the data needed for the query and four hours to recreate the query and section on the target environment. This is about 22,500 resource hours. Then if we give an average labor rate of $80 per hour, the total resource cost is $1,800,000.

It would take one administrator about thirty seconds or a minute to execute the script but we should allow two hours to configure the script and validate the results. We can also round up the execute time to 15 minutes. Therefore, the cost of this method is $172 resulting in a cost savings of $1,799,828 from a manual migration process.

SSA has 258,096 query sections in EPM. This is also counting development and test queries so the actual number of queries that need to be migrated may be much lower.

Conclusion

In this paper, I described technical methods that one can leverage to ease migration from Oracle's Hyperion Interactive Report solution also known as EPM within the SSA community. I walked through the EPM database and how meta data for the documents and sections are stored in the database tables. I provided example SQL queries on how one could query these tables to recreate the components of the document sections. This includes the table, joins along with the related display, sort, and filter fields. I also showed how you could partially recreate the result, import and table sections.

I discussed how one could then build on the database information to use restful web services to extract section content and recreate on a target environment, specifically in this case to WebFocus.

Lastly, I discussed a web application that we developed as a prototype to show how you can use the database to track and view document information and content.

Appendix

Appendix A – EPM Tables

EPM Database Tables
V8_BQ_SECTION
V8_CONT_VERSION
V8_CONTAINER
V8_DATA_CONTAINER
V8_DBACCESS
V8_DBCONNECT_PARMS
V8_DBSERVER
V8_DBSERVER_DB
V8_DBTYPE
V8_FILE_SYSTEM_URL
V8_FILE_TYPE

V8_FLAGS
V8_FOLDER
V8_FOLDER_TREE
V8_H_CONSTANT
V8_H_DB_COLUMN
V8_H_DB_DATABASE
V8_H_DB_OWNER
V8_H_DB_TABLE
V8_H_DM_COLUMN
V8_H_DM_DIGEST
V8_H_DM_JOIN
V8_H_DM_LEAF_COL
V8_H_DM_LIMIT_COL
V8_H_DM_META_DEP
V8_H_DM_SECTION
V8_H_DM_TABLE
V8_H_DOCUMENT
V8_H_HPSU_COLUMN
V8_H_HPSU_TABLE
V8_H_OBJECT_ID
V8_H_OLAP_DIM
V8_H_OLAP_FILTER
V8_H_OLAP_MBRSEL
V8_H_OLAP_NAMES
V8_H_OLAP_QUERY
V8_H_OLAP_SECTION
V8_H_QRY_COLUMN
V8_H_QRY_DM_REF
V8_H_QRY_QUERY
V8_H_QRYCOL_DMCOL
V8_H_QRYCOL_RSCOL
V8_H_RS_COLUMN
V8_H_RS_COMP_ITEM
V8_H_RS_LIMIT_COL
V8_H_SECTION
V8_H_SECTION_DEP
V8_H_TBL_COLUMN
V8_H_WA_DATASRC
V8_OCE
V8_OCE_FILE
V8_QRY_DB_CONN
V8_USER_ATTRIBUTES
V8_USER_VW
V8_USERROLEASSC

Appendix B – Code to Create DocumentContent and populate with section data

```
--create documentcontent table to hold section query parts
CREATE TABLE [dbo].[DocumentContent](
	[DocumentContent_id] [int] IDENTITY(1,1) NOT NULL,
	[DocumentID] [numeric](19, 0) NULL,
	[SectionID] [numeric](19, 0) NULL,
	[DataSourceID] [varchar](64) NULL,
	[QueryText] [varchar](max) NULL,
	[WebFocusText] [varchar](max) NULL,
	[TableauText] [varchar](max) NULL,
	[SSRSText] [varchar](max) NULL,
	[SectionName] [varchar](255) NULL,
	[DisplayFields] [varchar](max) NULL,
	[Tables] [varchar](max) NULL,
	[Joins] [varchar](max) NULL,
	[Filters] [varchar](max) NULL,
	[ComputedFields] [varchar](max) NULL,
	[SortFields] [varchar](max) NULL,
	[Section_Already_Migrated] [bit] NULL,
	[Section_Already_Migrated_By_Pin] [nvarchar](6) NULL,
	[Section_Already_Migrated_By_First] [nvarchar](50) NULL,
	[Section_Already_Migrated_By_Last] [nvarchar](50) NULL,
	[Section_Already_Migrated_Date] [date] NULL,
 CONSTRAINT [PK_DocumentContent] PRIMARY KEY CLUSTERED
([DocumentContent_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]


--populate document content table with document sections, this will insert all sections
insert into [dbo].[DocumentContent] ([DocumentID], [SectionID], [SectionName])
select d.document_id, s.section_id, s.SECTION_NAME from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.DOCUMENT_ID = s.DOCUMENT_ID
where section_id not in (select section_id from [dbo].[DocumentContent])
order by d.document_id, s.section_id

--update display fields just for one section 77748808 (remove and s.SECTION_ID = 77748808 to update all sections)
Update [dbo].[DocumentContent]
set [DisplayFields] = x.DisplayFields
from [dbo].[DocumentContent] dc inner join
(select x1.section_id,
 string_agg(cast( concat('[',x1.RDBMS_TABLE_NAME,'].[',x1.RDBMS_COLUMN_NAME,']') as varchar(max)), ', ') as
DisplayFields
 from (
 select distinct  s.SECTION_ID, dbt.RDBMS_TABLE_NAME, dbc1.RDBMS_COLUMN_NAME
 from [V8_H_DOCUMENT] d
 inner join [V8_H_SECTION] s on d.document_id = s.document_id
 inner join [V8_H_DM_TABLE] dmt  on s.SECTION_ID = dmt.SECTION_ID
inner join [V8_H_DB_TABLE] dbt on dmt.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
inner join [V8_H_DM_COLUMN] dmc1 on dmt.DM_TABLE_ID =dmc1.DM_TABLE_ID
inner join [V8_H_DB_COLUMN] dbc1 on dmc1.RDBMS_COLUMN_ID = dbc1.RDBMS_COLUMN_ID
 where SECTION_TYPE_ID = 4 and s.SECTION_ID = 77748808
 ) as x1
group by x1.SECTION_ID) as x on x.section_id = dc.sectionID

  --update tables just for one section 77748808 (remove and s.SECTION_ID = 77748808 to update all sections)
Update [dbo].[DocumentContent]
set [Tables] = x.Tables
from [dbo].[DocumentContent] dc inner join
(
select distinct dmt.section_id, string_agg(cast(concat('[',dbt.RDBMS_TABLE_NAME,']') as varchar(max)), ', ') as Tables
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join  [V8_H_DM_TABLE] dmt  on s.SECTION_ID = dmt.SECTION_ID
inner join  [V8_H_DB_TABLE] dbt on dmt.RDBMS_TABLE_ID = dbt.RDBMS_TABLE_ID
where SECTION_TYPE_ID = 4 and s.section_id = 77748808
group by dmt.section_id) as x on x.section_id = dc.sectionID

--update joins just for one section 77748808 (remove and s.SECTION_ID = 77748808 to update all sections)
Update [dbo].[DocumentContent]
set [Joins] = x.Joinclause
from [dbo].[DocumentContent] dc inner join
 (
select z.section_id,
string_agg(cast(concat(z.tbl1,'.',z.col1, ' = ', z.tbl2,'.', z.col2) as varchar(max)),' and ') as Joinclause
from (
select distinct s.section_id, dbt1.RDBMS_TABLE_NAME as tbl1, dbc1.RDBMS_COLUMN_NAME as col1,
dbt2.RDBMS_TABLE_NAME as tbl2, dbc2.RDBMS_COLUMN_NAME as col2
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_DM_JOIN] dmj on s.SECTION_ID = dmj.SECTION_ID
inner join [V8_H_DM_COLUMN] dmc1 on dmj.FROM_HPSU_COLUMN_ID = dmc1.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc1 on dmc1.RDBMS_COLUMN_ID = dbc1.RDBMS_COLUMN_ID
inner join [V8_H_DB_TABLE] dbt1 on dbc1.RDBMS_TABLE_ID = dbt1.RDBMS_TABLE_ID
inner join [V8_H_DM_COLUMN] dmc2 on dmj.TO_HPSU_COLUMN_ID = dmc2.DM_COLUMN_ID
inner join [V8_H_DB_COLUMN] dbc2 on dmc2.RDBMS_COLUMN_ID = dbc2.RDBMS_COLUMN_ID
inner join [V8_H_DB_TABLE] dbt2 on dbc2.RDBMS_TABLE_ID = dbt2.RDBMS_TABLE_ID
where SECTION_TYPE_ID = 4 and s.section_id = 77748808
) as z
group by z.SECTION_ID) as x
on x.section_id = dc.sectionID

--update filters Text
Update [dbo].[DocumentContent]
set [Filters] = x.Filters
from [dbo].[DocumentContent] dc inner join
(select s.section_id, cast(string_agg( dl.LIMIT_VALUE,' and ') as varchar(max)) as Filters
from [V8_H_DOCUMENT] d
inner join [V8_H_SECTION] s on d.document_id = s.document_id
inner join [V8_H_DM_LIMIT_COL] dl on dl.PARENT_SECTION_ID = s.SECTION_ID
where SECTION_TYPE_ID = 4 and s.section_id = 77748808
group by s.section_id) as x
on x.section_id = dc.sectionID
```

Appendix C – Python Script to upload file to WebFocus using Restful Web API

```
import pypyodbc
import csv
import sys
import pandas as pd
import requests
import base64

from requests.auth import AuthBase
from requests.auth import HTTPBasicAuth
from requests.auth import HTTPDigestAuth

import xml.etree.ElementTree as ET
from xml.dom import minidom
```

```
##define the api endpoint and user credentials

devuser = 'sampleuser'
devpw = 'samplepassword'
strbaseurl = 'http://<webserver name>/ibi_apps/rs/ibfs' #change to actual webfocus URL
wfDevURL = strbaseurl + '?IBIRS_action=signOn&IBIRS_userName=' + devuser + "&IBIRS_password=" + devpw

#define folder to be migrated
mgfolder =  '/BI Central Office Folders/DCBFM/DCBFM FPA/Development/migrationScript_test'

getdocsSQL = "select document_id from [V8_H_DOCUMENT] where path = '" + mgfolder + "'"

#define function that when called, does all the work needed
def signonWF():
    response = requests.post(wfDevURL)
    status = response.status_code

    if status == 200:
        print('successfully connected ', response.status_code)

        r = response.text
        #r.write('output.xml')
        cookie = response.cookies
        #print(cookie)

        root = ET.fromstring(r)

        tree = ET.ElementTree(root)
        #write to xml file so that minidom can parse it
        tree.write("output.xml")

        dom = minidom.parse("output.xml")
        xmltags = dom.getElementsByTagName('entry')
        stoken = [items.attributes['value'].value for items in xmltags if items.attributes['key'].value ==
"IBI_CSRF_Token_Value"]
        print('security token is ', stoken)

        mdocs = pd.read_sql(getdocsSQL,con=conn)
        d = 0
        for row in mdocs.values:
            docid = row[0]
            #identify all query sections that have webfocus content for each document
            secsql = "SELECT concat(replace(d.DOCUMENT,' ','_'), '_sec_', replace(s.SECTION_NAME,' ','_')) as docTitle, \
            [WebFocusText] FROM [dbo].[DocumentContent] dc  inner join [V8_H_DOCUMENT] d on dc.DocumentID =
d.DOCUMENT_ID \
            inner join [V8_H_SECTION] s on dc.SectionID = s.SECTION_ID where not [WebFocusText] is null and not
WebFocusText = 'NO DATA' \
            and s.SECTION_TYPE_ID = 8 and dc.documentid = " + str(docid)

            wfData = pd.read_sql(secsql,con=conn)

            i = 0
            for row in wfData.values:

                #get report title
                description = row[0]
                filecontent = row[1]

                byteobj = base64.b64encode(filecontent.encode())
                fileobj = byteobj.decode("utf-8")
                #print(fileobj)
                reportpath = '/WFC/Repository/BIMigration/uploadtest/' + description + '.fex'
                strtoken = stoken[0]

                secToken = '&IBIWF_SES_AUTH_TOKEN=' + strtoken
                putObj = '<rootObject _jt="IBFSMRObject" description="' + description + '" type="FexFile"> <content
_jt="IBFSByteContent" char_set="Cp1252">' + fileobj + '</content> </rootObject>'
                upURL = strbaseurl + reportpath + '?IBIRS_action=put&IBIRS_object=' + putObj + secToken
                #print(upURL)

                uploadresponse = requests.post(upURL, cookies=cookie)
                #file.close
                #print(uploadresponse.text)
                #increment for next section
                i+=1

            #increment for next document
            d+=1

    else:
        print('Not able to connect to webfocus, the status code is ' + status)


##connection string to migration database
conn = pypyodbc.connect("DRIVER={SQL Server};"
                    "SERVER=<database server>;"
                    "DATABASE=<database name>;"
                    "Trusted_Connection=yes;")

cursor = conn.cursor()

#call function to get documents and perform migration
signonWF()

#cleanup database connection
conn.commit()
cursor.close()
conn.close()

print("all done migrating for this folder, please check the webfocus server")
```