u.achieve Data Analysis Outline

Skip to end of metadata

-
- Added by Dale Peters, last edited by Dale Peters on Sep 12, 2012 (view change)

Go to start of metadata

Valuable information about students' academic progress is contained in the audit result tables. This is a companion outline to u.achieve Data Analysis and provides a bullet overview of it's contents.

Although this document is written using the u.achieve version of the tables, all of the information also applies to DARwin except where noted below.

Background

Com Options

EXTOUT.

- By default, EXTOUT is not set. In this mode, data will be written to the job_queue_run, job_queue_out, and job_queue_report tables with each audit.
- To get data written to the other job_queue tables, one of the following EXTOUT options must be set:
  - **V** builds and writes output to five tables -
    - job_queue_req
    - job_queue_req_text
    - job_queue_subreq
    - job_queue_subreq_text
    - job_queue_course
  - **X** builds and writes output to three tables -
    - job_queue_req
    - job_queue_subreq
    - job_queue_course
  - **Y** builds and writes output to four tables -
    - job_queue_req

        - job_queue_subreq
        - job_queue_course
        - *job_queue_accept*
    - **Z** builds and writes output to seven tables -
        - job_queue_req
        - job_queue_req_text
        - job_queue_subreq
        - job_queue_subreq_text
        - job_queue_course
        - *job_queue_accept*
        - job_queue_markers
- Writing to job_queue_accept creates A LOT of data

Audit Request Tables

These tables are described in detail at u.achieve Audit Request Tables.

job_queue_list:

- Each row in this table represents one audit run.
- A run could be a single audit or a batch.
- Each run has its own unique jobid to identify the run.
- All audits in the run share the same jobid.
- Columns of interest:
    - jobid (mentioned later)
    - status
    - servername

job_queue_detail

- Child table of job_queue_list.
- Each row represents a request for an individual student.
- Batch runs are accomplished by inserting a row in this table for each student to be included in the batch, all associated to the same jobid.
- Columns of interest:
    - stuno
    - fdprog
    - fcatlyt

job_queue_sysin

- Child table of job_queue_list, alternative to job_queue_detail.
- Provides the ability to pass student data with the request.
- Data must be formatted within 80 character fields.

Making an Audit Request

Any process that can write to the audit request tables is able to make an audit request and have the request processed by u.achieve.

- Follow the data element rules described in u.achieve Audit Request Tables.
  - Audit request process ***must*** create a **unique** JobID.
- The u.achieve server will start processing a job as soon as it finds a new row in job_queue_list with a status of 'N' and a matching servername, so watch out for...
  - servername not matching
  - jobs that start processing too soon.

Typical Audit Request

```
Job_Queue_List

instidq     instid     instcd     jobid             userid                          status     priority     startdate
starttime     external_eval     read_sysin     servername     last_mod_user        last_mod_date          comkey
log_level     report_type     ignore_cache
----------  ---------  ---------  ----------------  ------------------------------  ---------  -----------  -----------
-  ------------  ----------------  -------------  -------------  -------------------  ----------------------  ---------
------------  --------------  ---------------
73          TEST40                1S00009338554059  audit_test                       N          1
(null)            (null)         u.ach          dw40                3/21/2012 4:05:16 PM   ONL        D
(null)          (null)

Job_Queue_Detail

int_seq_no     instidq     instid     instcd     jobid             userid                          user_seq_no
stuno          comkey     evalsw     f1or2     report     pntreq     listall     test     fdprog          fcatlyt
sinstid     sinstcd     fdpmask          time_token     lasertext     format     parseflg     revart     binstid
binstcd     bytaken     soprid     status     d_whatif
-------------  ----------  ---------  ---------  ----------------  ------------------------------  --------------  ----
---------  ---------  ---------  --------  ---------  ---------  ----------  -------  ---------------  ----------  ----
------  ----------  --------------  -------------  ------------  ---------  -----------  ---------  ----------  ------
----  ----------  ---------  ---------  -----------
42675          73          TEST40                1S00009338554059  audit_test                      0
91606          40TST                 (null)
(null)
```

Audit Output Tables

See u.achieve Output Tables for a full description of all the output tables.

Job_Queue_Run

- One row for each audit run
- If multiple audits were included in a batch run, each resulting audit report gets its own row in job_queue_run entry.

Job_Queue_Req

- One row for each requirement included in the audit (both hidden and viewable requirements are included)

Job_Queue_Req_Text

- Requirement header and title text

Job_Queue_SubReq

- One row for each sub-requirement in the audit.

Job_Queue_SubReq_Text

- Sub-Requirement header and title text

Job_Queue_Course

- Child table of job_queue_subreq
- Student courses that used by the sub-requirement

Job_Queue_Accept

- Child table of job_queue_subreq
- Select From / Reject courses for the sub-requirement
- *produces A LOT of data*

Most of the data analysis you'll be doing will be concerned with the audit output tables. A few notes about the data you'll find (or won't find) in those tables:

- You might get **more** audits than you think you requested.
  - if the student's degree program is not specified as part of the audit request AND the student is enrolled in more than one program. By default, each program will be run in that situation.
- You might get **fewer** audits than you requested
  - if you're running with one of the COM.NOREPORT options.
- Every audit that is run will get a row in job_queue_run.
  - If a single audit request generates two (or more) audit runs for the student, each separate report will get it's own job_queue_run row (except as affected by COM.NOREPORT).

Identifying Audit Results

Your audit results table is likely full of audits requested by students, faculty, and staff.

- Many of these audits could be duplicates.
- Some of these audits could be "what-if" audits.
- Some might not even be from your own students (depending on your use of u.select and its configuration).

The first step in analyzing your audit data is to determine which audits you want to include in your analysis.

By JobID

Single Batch

Every audit in the output tables will get the JobID value from its audit request.

> If all the audits that you're interested in can be included in a single batch run (one job_queue_list row, many job_queue_detail rows), then identifying those rows in the output tables is easily achieved by using the JobID value.

Multiple Batches

Use a JobID pattern to identify audits

If audit requests must be broken into several batches, then we recommend creating a JobID pattern that will be unique for the set of batches that you're interested in.

For example, you might use date + Batch ID + batch number:

-
  - 20120614GRADCHK1,
  - 20120614GRADCHK2,
  - etc.

Any combination of characters within 16 characters that makes the JobID unique is valid.

Other Methods

Other methods of identifying audits results that belong together may be less reliable unless you have some strong external controls in place.

servername

If you direct all of your analytic runs to a specific daemon servername, and you're sure 'old' duplicate audits aren't remaining in the tables or you use the rundate to filter results, then this could be a reasonable option.

rundate

Rundate will likely only be a reasonable option when used in combination with other columns to determine the set of output that you're interested in.

Analyzing Audit Results

Graduation Check

We can quickly determine if an audit is complete by looking at the "complete" and "IP" columns of the Job_Queue_Run table.

"Complete" Value	Meaning	"IP" Value	Meaning
<blank>	All requirements NOT met	<blank>	No In-Progress or Planned (what-if) courses used
C	Complete - All Requirements met	I	In-Progress courses used
E	Error - Error during audit processing	W	Planned (What-if) courses used.

When both planned and in-progress courses are used, 'W' is set.

Find Audits That Are Complete

```
select stuno, dprog, catlyt, complete, ip
from job_queue_run
where jobid like '201205%'
and complete = 'C'
```

Results

stuno	dprog	catlyt	complete	ip
11736	11736	200101	C	
12272	12272	201001	C	
13149	13149	201003	C	I
1773	1773	200101	C	
1966	1966	200101	C	
1988	1988	200101	C	
9-X2	BS-TEST	200408	C	W
ANA	TRLOOKUP	201003	C	
CCCHOMER	CCCHOMER	200808	C	W
COURSELIST	COURSELIST	000000	C	
MUPPET	MUPCOURSES	2012	C	
OWL-11906	OWL-11906	201008	C	
OWL-12209	OWL-12209	201009	C	
OWL-13051	OWL-13051	201208	C	
OWL11493	OWL11493	000000	C	I
TEST	HUNGRYSOLO	200101	C	
UACH1803	UACH1803	200101	C	
UACH2061	UACH2061	20121	C	
UCLA-TA1	TRLKP1	201201	C	
VARR2	VARR	201001	C	
VARR3	VARR	201001	C	

By adding a parameter to the query we could filter out those programs that are complete with in-progress or planned courses.

Find Audits That Are Complete Where No In-Progress or Planned Courses Are Used

```
select stuno, dprog, catlyt, complete, ip
from job_queue_run
where jobid like '201205%'
and complete = 'C'
and ip = ' '
```

For some purposes, maybe "complete" using in-progress courses is valid, but we do not want to include audits that are complete with planned courses.

Find Audits That Are Complete Where No Planned Courses Are Used

```
select stuno, dprog, catlyt, complete, ip
from job_queue_run
where jobid like '201205%'
and complete = 'C'
and ip != 'W'
```

Results

stuno	dprog	catlyt	complete	ip
11736	11736	200101	C	
12272	12272	201001	C	
13149	13149	201003	C	I
1773	1773	200101	C	
1966	1966	200101	C	
1988	1988	200101	C	
ANA	TRLOOKUP	201003	C	
COURSELIST	COURSELIST	000000	C	
MUPPET	MUPCOURSES	2012	C	
OWL-11906	OWL-11906	201008	C	
OWL-12209	OWL-12209	201009	C	
OWL-13051	OWL-13051	201208	C	
OWL11493	OWL11493	000000	C	I
TEST	HUNGRYSOLO	200101	C	
UACH1803	UACH1803	200101	C	
UACH2061	UACH2061	20121	C	
UCLA-TA1	TRLKP1	201201	C	
VARR2	VARR	201001	C	
VARR3	VARR	201001	C	

(A better resolution to the above scenario would be to not include any planned courses in the audit run in the first place, but to be safe it would never hurt to include the clause `ip != 'W'`)

In many cases, it might be more interesting to see the student's audits that do NOT have all degree requirements completed. For our case, let's say we want a list of student programs that are not completely finished - in-progress and planned courses should not be used.

Find All Audits That Are NOT Complete, or Only Complete Using IP or Planned Courses

```
select stuno, dprog, catlyt, complete, ip
from job_queue_run
where jobid like '201205%'
and (complete = ' ' or (complete = 'C' and ip != ' '))
```

Results

stuno	dprog	catlyt	complete	ip
12742	EVERYTHING	200808		
CCCHOMER	CCCHOMER	200808	C	W

```
12729          12729           201001
ADDCT0         ADDCT0          200101
ALLOWSPLIT     ALLOWSPLIT      200101
ALLOWSPLIT     ALLOWSPLIT2     200101
CLINE/HRSOUTU  CLINE/HRSOUTU   200101     C            I
CONDREQ        CONDREQ         000000                  I
```

Once we have this set of students identified, it might be interesting to figure out *why* these audits are not complete. That will require use to explore data in the requirement, sub-requirement, course, and accept tables.

Querying Requirement Data

A wealth of information is available to mine at the requirement level of the Audit Output Tables. The tricky part is figuring out how to get at it. The relationship between the job_queue_run and job_queue_requirement table is:

```
job_queue_run.int_seq_no = job_queue_req.jobq_seq_no
```

In a typical query (with an implied INNER JOIN) you would define the relationship as:

'Starter Query' for job_queue_req data

```
select run.stuno, run.dprog, run.catlyt, req.*
from job_queue_req req, job_queue_run run
where run.int_seq_no = req.jobq_seq_no
and run.jobid = ...

(This query is incomplete, but you just need to build up the where clause to run it.
The table relationships are complete.)
```

Assuming we have a handle on the set of audits we're interested in, narrowing our scope to a specific set of requirements will require us to use one of three criteria.

By Requirement Name (rname)

If we have interest in a specific requirement that may be in use across all audits, then we can simply search for the requirement by name. For sake of interest, we'll look for all cases where the total hour requirement is not met. In our example the name of our total hour requirement is 'THOUR2'.

Requirements By Name, Not Met

```
select run.stuno, run.dprog, run.catlyt, req.rname, req.reqgpa, req.gotgpa, req.reqhrs, req.gothrs, req.reqct, req.gotct
from job_queue_req req, job_queue_run run
where req.jobq_seq_no = run.int_seq_no
and run.jobid = '201205%'
```

```
and req.satisfied = 'N'
and rname = 'THOUR2'
order by run.stuno
```

stuno	dprog	catlyt	rname	reqgpa	gotgpa	reqhrs	gothrs	reqct	gotct
EXC1	EXCEPT1	200408	THOUR2	3	2.5	0	6	0	2
EXC1A	EXCEPT1	200408	THOUR2	3	1.2	0	9	0	3
EXC1B	EXCEPT1	200408	THOUR2	3	1.6	0	15	0	5
EXC2	EXCEPT2	200408	THOUR2	3	2.2	0	15	0	5
EXC4	EXCEPT4	200408	THOUR2	3	0	0	0	0	0
EXC4A	EXCEPT4	200408	THOUR2	3	0	0	0	0	0
EXC5A	EXCEPT5	200408	THOUR2	3	2.666	0	9	0	3
EXC5E	EXCEPT9	200408	THOUR2	3	2.75	0	24	0	8
EXC5G	EXCEPT5	200408	THOUR2	3	2.666	0	9	0	3
EXC5L	EXCEPT9	200408	THOUR2	3	2.75	0	24	0	8
EXC7B	EXCEPT7	200408	THOUR2	3	1.6	0	10	0	3
EXC7C	EXCEPT7	200408	THOUR2	3	2	0	3	0	1
EXC8	EXCEPT8	200408	THOUR2	3	2.8	0	15	0	5

If we care to know WHY a requirement is not satisfied, we could add more criteria to find out.

Requirements By Name, GPA Not Met

```
select run.stuno, run.dprog, run.catlyt, req.rname, req.reqgpa, req.gotgpa, req.reqhrs, req.gothrs, req.reqct, req.gotct
from job_queue_req req, job_queue_run run
where req.jobq_seq_no = run.int_seq_no
and run.jobid = '201205%'
req.satisfied = 'N'
and (req.reqgpa < req.gotgpa)
and rname = 'THOUR2'
order by run.stuno
```

Meaningful requirements names are great!!

but aren't always possible

- Naming conventions started years (decades?) ago
- Often not able to use the same 'common' requirement across all programs
- Different programs may have different requirement definitions to achieve the same thing
    - i.e. total hours requirement for program A uses TOTHR120, program uses TOTHR128)

There are still options available to find the data in a single query across different programs.

By Category

We could also use an Interactive Audit Chart Category as a search criteria parameter.

> Encoding for the Interactive Audit

This is fundamentally different than searching by requirement name in a few key ways:

- The same category names are typically used across all programs
  - "Major" category will be found in all of your programs, even though the specific requirements that make up the "Major" category will be different in each program.
- Within a program, a single category will likely have many requirements associated with it.
  - If you're looking for Major GPA, but there are several requirements associated with the Major category in a single program, you won't be able to select the exact GPA you want by using only the category column.

Each category can also have a specific set of properties associated with it that can be useful when mining data, for example, total_hour and total_gpa.

Total Hour Requirement Not Met, Find By total_hour

```
select distinct run.stuno, run.dprog, run.catlyt, req.rname, req.reqhrs, req.gothrs, req.total_hour
from job_queue_req req, job_queue_run run
where req.jobq_seq_no = run.int_seq_no
and run.jobid < '201206'
and satisfied = 'N'
and req.total_hour = 'T'
order by run.stuno
```

Notice we're getting results across different requirements (RName values).

Result

stuno	dprog	catlyt	rname	reqhrs	gothrs	total_hour
ALTCATGUY	PSYCH BA	20048	TOT-HOUR	128	2	T
ALTCATGUY2	PSYCH BA	20048	TOT-HOUR	128	6	T
ALTCATGUY3	PSYCH BA	20048	TOT-HOUR	128	9	T
HOMEY	GENERAL-CWS	201008	TOT-HOUR	128	3	T
REQFLAGS	REQFLAGS-REQ	20121	TOTAL-HRS	120	29	T

By GpaName

A less frequently used option is the GpaName field of the requirement. This field is available for the sole purpose of identifying a requirement in the output tables so that it can be easily retrieved later. The original intent of the field, as evidenced by its name, was to clearly and easily identify the

requirement within a program that was associated with a GPA calculation, so that those calculated GPA could be retrieved after the audit results are saved. But the field GpaName can be used to label a requirement for any purpose.

It's similar to Category in that:

- The use of GpaName does not affect audit processing in any way.
- The same GpaName can (and should) be used in many requirements

It's different from Category in that:

- A GpaName should likely only show up once within a program. An exception would be that a dual major program could reasonalbly have two Major Gpa values that you could identify with the same "MajorGpa" GpaName value.

A typical example would be to label all Major Requirements with the same GpaName, even though the actual Requirement Names (RName) are completely different. Then one can easily query the status of all students in their major requirement, regardless of the requirement name or the programs they are in.

Display Major GPA for All Students

```
select run.stuno, run.dprog, run.catlyt, req.rname, req.gpaname, req.gotgpa
from job_queue_req req, job_queue_run run
where req.jobq_seq_no = run.int_seq_no
and run.jobid = '2012061114194835'
and run.rundate > '06/13/2012'
and req.gpaname = 'MAJORREQ'
```

Results

stuno	dprog	catlyt	rname	gpaname	gotgpa
03880103	B1FSFDST	20121	ANRFDST-M	MAJORREQ	3.91
00170730	BSBABSAD	20111	CBABSAD	MAJORREQ	3.33
03957275	BSBAMRKT	20091	CBAMRKT	MAJORREQ	2.832
03880103	BA ENGL	20081	ASCENGL	MAJORREQ	3.967
23993775	B1WSWATR	20101	ANRWATR-C	MAJORREQ	3.837

End Part I

Interesting ? Queries on Requirement Data

With these methods for identifying requirements in place, we can run some interesting queries.

Using Estimated Required Hours

Assuming our goal is to determine each student's major requirement percent complete, you might be first inclined to compare GotHours to Required Hours, or GotCount to Required Count, or maybe GotSubReqs to Required SubReqs to determine percent complete. But some requirements might require hours and others might require count. Even others might not require either at the requirement level and defer those comparisons to the sub-requirement level. It would be cumbersome to write a single query that would handle the calculation no matter which field(s) are used.

No worries, estimated required hours to the rescue!

Estimated Required Hours will be calculated for EVERY requirement and sub-requirement no matter what is actually required. It is a (fairly accurate) estimation of the total number of hours of coursework that the student must take to complete the requirement or sub-requirement. The value of estimated hours might change up or down as the student takes courses, exceptions are applied, limits are enforced, etc. By using estimated hours we can get develop a relatively simple query that gives us very powerful results.

Major Requirements Percent Complete

```
select run.stuno, run.dprog, run.catlyt, req.rname, req.gpaname, req.est_reqhrs, req.gothrs, 100*(gothrs/req.est_reqhrs)
from job_queue_req req, job_queue_run run
where req.jobq_seq_no = run.int_seq_no
and run.jobid = '2012061114194835'
and run.rundate > '06/13/2012'
and req.gpaname = 'MAJORREQ'
and run.report_type = ' '
```

Proper SQL code would ensure we don't encounter a divide by zero error, and that the percent calculation is rounded and formatted properly. But this isn't a SQL lesson...

Results

stuno	dprog	catlyt	rname	gpaname	est_reqhrs	gothrs	column8
03880103	B1FSFDST	20121	ANRFDST-M	MAJORREQ	53	11	20.75471698113207
00170730	BSBABSAD	20111	CBABSAD	MAJORREQ	21	6	28.57142857142857
03957275	BSBAMRKT	20091	CBAMRKT	MAJORREQ	18	18	100
03880103	BA ENGL	20081	ASCENGL	MAJORREQ	36	36	100
23993775	B1WSWATR	20101	ANRWATR-C	MAJORREQ	73	66	90.41095890410958

Querying Sub-Requirement Data

The data available at the sub-requirement level is very similar to the requirement data. A typical query is setup with the table relationships as:

'Starter Query' for job_queue_subreq data

```
select run.stuno, run.dprog, run.catlyt, subreq.*
from job_queue_subreq subreq, job_queue_req req, job_queue_run run
where run.int_seq_no = req.jobq_seq_no
and subreq.jobq_seq_no = req.jobq_seq_no
```

```
and run.jobid = ...

(This query is incomplete, but you just need to build up the where clause to run it.
The table relationships are complete.)
```

With only a few modifications to look at data from the sub-requirement table, we can run a similar "percent complete" query.

SubRequirements Percent Complete from UnMet Reqs that are Not Complete

```
select run.stuno, run.dprog, run.catlyt, req.rname, req.gpaname, subreq.stabx, subreq.est_reqhrs, subreq.gothrs,
100*(subreq.gothrs/subreq.est_reqhrs)
from job_queue_subreq subreq, job_queue_req req, job_queue_run run
where req.jobq_seq_no = run.int_seq_no
and subreq.jobq_seq_no = req.jobq_seq_no
and run.jobid = '2012061114194835'
and run.rundate > '06/13/2012'
and req.gpaname = 'MAJORREQ'
and run.report_type = ' '
and subreq.est_reqhrs > 0
and subreq.gothrs < subreq.est_reqhrs
and req.satisfied = 'N'
```

Results

```
 stuno          dprog            catlyt     rname      gpaname          stabx     est_reqhrs     gothrs     column9
 -------------  ---------------  ---------  ---------  ---------------  --------  -------------  ---------  -----------
------
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         082       4              0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         083       7              4
57.14285714285714
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         085       12             0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         086       5              4          80
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         088       3              0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         095       3              0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         102       12             0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         103       12             0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         104       10             0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         105       5              0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         106       3              0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         108       300            117        39
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         109       300            0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         110       300            0          0
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         111       300            0          0
 00170730       BSBABSAD         20111      CBABSAD    MAJORREQ         074       9              1
11.11111111111111
 00170730       BSBABSAD         20111      CBABSAD    MAJORREQ         075       300            0          0
 00170730       BSBABSAD         20111      CBABSAD    MAJORREQ         076       120            95
79.16666666666666
 00170730       BSBABSAD         20111      CBABSAD    MAJORREQ         078       3              0          0
```

```
 00170730       BSBABSAD         20111       CBABSAD     MAJORREQ          091       18              3
16.66666666666666
```

All the audit text is also available to us, which might make this query more meaningful. Adding the table relationships makes the query look more complicated, but its rather straightforward. We'll simply add some text to our percent complete query to make it more readable. To do so, we need to define the text relationship as an OUTER JOIN, which means we want to display the data even though title text might not be defined for each subreq. You'll notice the style of the table relationships will change.

Sub-Requirement Percent Complete, with Titles

```
select run.stuno, run.dprog, run.catlyt, req.rname, req.gpaname, subreq.stabx, subreq.est_reqhrs, subreq.gothrs,
100*(subreq.gothrs/subreq.est_reqhrs), srtext.text
from job_queue_run run
inner join job_queue_req req on req.jobq_seq_no = run.int_seq_no
inner join job_queue_subreq subreq on subreq.jobq_seq_no = run.int_seq_no
left outer join job_queue_subreq_text srtext on srtext.jobq_seq_no = run.int_seq_no and srtext.rtabx = req.rtabx
where run.jobid = '2012061114194835'
and run.rundate > '06/13/2012'
and req.gpaname = 'MAJORREQ'
and run.report_type = ' '
and subreq.est_reqhrs > 0
and subreq.gothrs < subreq.est_reqhrs
and req.satisfied = 'N'
```

Querying Student Course Data

The job_queue_course table holds all the student courses that matched a sub-requirement. *Note that a student's course will be replicated in the job_queue_course table every time it matches a different sub-requirement.* This means that a student with 20 courses could easily have well over 200 entries in the job_queue_course table for a single audit run.

We'll use the inner/outer join query style so that sub-requirements will continue to display even when no student courses match.

Sub-Requirement % Complete, with Courses

```
select run.stuno, run.dprog, run.catlyt, req.rname, req.gpaname, subreq.stabx, subreq.est_reqhrs, subreq.gothrs,
100*(subreq.gothrs/subreq.est_reqhrs) as '% complete', course.yt, course.course
from job_queue_run run
inner join job_queue_req req on req.jobq_seq_no = run.int_seq_no
inner join job_queue_subreq subreq on subreq.jobq_seq_no = run.int_seq_no
left outer join job_queue_course course on course.jobq_seq_no = run.int_seq_no and course.rtabx = subreq.rtabx and
course.stabx = subreq.stabx
where run.jobid = '2012061114194835'
and run.rundate > '06/13/2012'
and req.gpaname = 'MAJORREQ'
```

```
and run.report_type = ' '
and subreq.est_reqhrs > 0
and subreq.gothrs < subreq.est_reqhrs
and req.satisfied = 'N'
```

results

```
 stuno          dprog            catlyt     rname      gpaname          stabx     est_reqhrs     gothrs     % complete
yt      course
 -------------  ---------------  ---------  ---------  ---------------  --------  -------------  ---------  -----------
------  ------  ---------------
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         082       4              0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         083       7              4
57.14285714285714  20082   BIOS101
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         083       7              4
57.14285714285714  20082   BIOS101L
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         085       12             0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         086       5              4          80
20121   MSYM109
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         088       3              0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         095       3              0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         102       12             0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         103       12             0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         104       10             0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         105       5              0          0
(null)  (null)
 03880103       B1FSFDST         20121      ANRFDST-M  MAJORREQ         106       3              0          0
(null)  (null)
...
```

Aggregate queries on student course data might also be useful. We must be careful that we don't count a coure more than once within the same requirement. Major requirements will likely have some hidden sub-requirements that might set condition codes and likely have a reuse option set.

Let's count how many times each course is used in *any* major requirement. Notice that we're using GPANAME to identify those requirements that we consider to be 'Major Requirements', regardless of the actual value of the Requirement Name (RName).

Count of Courses Used in Major Requirements

```
select course.course, count (distinct run.stuno + req.rname + course.course) as courseCount
from job_queue_run run
inner join job_queue_req req on req.jobq_seq_no = run.int_seq_no
```

```
inner join job_queue_subreq subreq on subreq.jobq_seq_no = run.int_seq_no
left outer join job_queue_course course on course.jobq_seq_no = run.int_seq_no and course.rtabx = subreq.rtabx and
course.stabx = subreq.stabx
where run.jobid = '2012061114194835'
and run.rundate > '06/13/2012'
and req.gpaname = 'MAJORREQ'
and run.report_type = ' '
group by course
order by courseCount desc
```

Results

course	courseCount
ECON212	5
ENGL150	5
MATH101	3
MATH102	3
COMM109	3
AECN265	3
AGRO442	3
BIOS112	3
BIOS112L	3
BIOS101	3
NRES220	3
NRES222	3
NRES312	3
NRES323	3
NRES450	3
NRES474	3
NRES498	3
STAT218	3
NRES433	2
NRES497	2
UHON395H	2
POLS272	2
ENGL045@	2
NRES311H	2
MUSC189H	2
NRES103	2
NRES211	2
HSPN337	2
JGEN220	2
LIBR110	2
MSYM109	2
MUCO247	2
MUCO447	2
MUDC247	2
MUNM100D	2

```
MUNM287          2
HNAT338          2
```

Querying Accept Course Data

To get u.achieve to output data to the job_queue_accept table, the Com field EXTOUT must be set to 'Y'. This will produce A LOT OF DATA, so you should only use this option when you're sure you plan to analyze the job_queue_accept data. Here's what I mean by 'A LOT OF DATA':

Estimate of Data Generated Using EXTOUT = Y

```
Encoding estimates:

50 requirements / degree program
 5 sub-requirements / requirement = 50 x 5 = 250 sub-requirements / program
15 courses / sub-requirement = 50 x 5 x 15 = 3750 courses/program

= 4050 rows of data / audit

------

x  5,000 students = 20,250,000 rows of data / full student batch
x 10,000 students = 40,500,000 rows of data / full student batch
x 15,000 students = 60,750,000 rows of data / full student batch
x 20,000 students = 81,000,000 rows of data / full student batch
```

The relationship of job_queue_accept to other tables is identical to the job_queue_course table relationships. In our sample query, we'll find which courses are available to be taken most often in major requirements. We should make sure we're not including 'reject' courses in our count.

```
select accept.course, count (distinct run.stuno + req.rname + accept.course) as courseCount
from job_queue_run run
inner join job_queue_req req on req.jobq_seq_no = run.int_seq_no
inner join job_queue_subreq subreq on subreq.jobq_seq_no = run.int_seq_no
inner join job_queue_accept accept on accept.jobq_seq_no = run.int_seq_no and accept.rtabx = subreq.rtabx and accept.stabx
= subreq.stabx
where run.jobid = '2012061114194835'
and run.rundate > '06/13/2012'
and req.gpaname = 'MAJORREQ'
and run.report_type = ' '
and course.reject != '1'
group by course
order by courseCount desc
```

Results

```
course           courseCount
 ---------------  --------------
 SOCI310B         5
```

```
ENGL245J        5
HIST202         5
ANTH291         5
MODL189H        5
ENGL210T        5
HENG****        5
HORT488         5
HIST****        5
ENGL311G        5
ENTO309         5
GEOL105         5
MUED450         5
ENTR291         5
ANTH242         5
CEHS200         5
NRES299         5
CSCE230         5
GEOG155         5
```

Like Be the first to like this

- None
- Edit Labels