How I Learned to use the Skyserver

Robert Lupton

Princeton, July 2002
I like to access the skyserver using my emacs-lisp mode; the latest version is v1.10 and is available from

http://www.astro.princeton.edu/~rhl/skyserver

There’s even a manual. Warning: older versions don’t support the full use of @variables.

You may prefer to use some other interface; in particular Jim Gray and Alex Szalay like the Microsoft Query Analyser which supports full Transact-SQL.
A Simple Query

```sql
select distinct
    run, rerun,
    field0, field0 + nfields - 1,
    photoVersion
from Segment
```
```plaintext
run rerun field0  photoVersion

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>94</td>
<td>7</td>
<td>104</td>
<td>544</td>
<td>v5_2_21</td>
</tr>
<tr>
<td>125</td>
<td>7</td>
<td>11</td>
<td>451</td>
<td>v5_2_21</td>
</tr>
<tr>
<td>752</td>
<td>8</td>
<td>11</td>
<td>617</td>
<td>v5_2_5</td>
</tr>
<tr>
<td>756</td>
<td>8</td>
<td>196</td>
<td>802</td>
<td>v5_2_6</td>
</tr>
<tr>
<td>1336</td>
<td>2</td>
<td>11</td>
<td>95</td>
<td>v5_2_12</td>
</tr>
<tr>
<td>1339</td>
<td>2</td>
<td>11</td>
<td>95</td>
<td>v5_2_12</td>
</tr>
<tr>
<td>1356</td>
<td>2</td>
<td>20</td>
<td>113</td>
<td>v5_2_12</td>
</tr>
<tr>
<td>1359</td>
<td>3</td>
<td>19</td>
<td>113</td>
<td>v5_2_12</td>
</tr>
</tbody>
</table>
```

Note the ugly formatting, and that the last field (the one calculated as `field0 + nfields - 1`) has no column heading.
A Formatted Simple Query

```sql
select distinct
    str(run, 4) as run, str(rerun, 3) as rerun,
    str(field0,3) as field0, str(field0 + nfields - 1,4) as field1,
    photoVersion
from Segment
order by run
```
Prettier and field1 is no longer anonymous. We’ve made the ordering by run explicit.
A Simple Egocentric Query

-- What’s loaded in skyserver?
--
declare @database char  set @database = Robert
select distinct
    str(run, 4) as run, str(rerun, 3) as rerun,
    str(field0,3) as field0, str(field0 + nfields - 1,4) as field1,
    photoVersion
from @database..Segment
order by run
The same as before, except that this time I wanted to know what was loaded in my database.
An Almost-as-Simple Query

-- What’s loaded in skyserver?

declare @database char set @database =

select distinct top 2
    str(run, 4) as run, str(rerun, 3) as rerun,
    str(field0,3) as field0, str(field0 + nfields - 1,4) as field1,
    photoVersion
from @database..Segment
where run != 745
order by run
```
run rerun field0 field1 photoVersion
  94  7  104  544 v5_2_21
 125  7  11  451 v5_2_21
```

Omit run 745, and only show the first two rows found by the query. We’re interrogating the main (non-Robert) database this time.
The Galaxy Target Selection Algorithm

declare @database char set @database =
declare @pi float set @pi = 3.141592654
__
declare @BLENDED int set @BLENDED = dbo.fPhotoFlags('BLENDED')
declare @BRIGHT int set @BRIGHT = dbo.fPhotoFlags('BRIGHT')
declare @EDGE int set @EDGE = dbo.fPhotoFlags('EDGE')
declare @NODEBLEND int set @NODEBLEND = dbo.fPhotoFlags('NODEBLEND')
declare @SATURATED int set @SATURATED = dbo.fPhotoFlags('SATURATED')
__
declare @bad_flags int set @bad_flags = (@SATURATED | @BRIGHT | @EDGE)
__
declare @maglim float set @maglim = 17.77
declare @SBlim float set @SBlim = 24.5
declare @delta_psf_model float set @delta_psf_model = 0.3
__
select top 10
-- Standard fields
run, rerun, camCol, field,
str(rowc,6,1) as rowc, str(colc,6,1) as colc,
str(dbo.fObjFromObjID(ObjId), 4) as id,
'::' as '::',
-- Scientific output
--
str(ra,9,4) as ra, str(dec,8,4) as dec
__
from
from @database..PhotoPrimary
where -- Our star-galaxy separation and target selection
    psfMag_r - modelMag_r >= @delta_psf_model and
    petroMag_r - reddening_r <= @maglim and
    petroMag_r - 2.5*log10(2*pi*petroR50_r*petroR50_r) < @SBlim and
-- Check flags
    (flags & @bad_flags) = 0 and
    (((flags & @BLENDED) = 0) or ((flags & @NODEBLEND) != 0))
<table>
<thead>
<tr>
<th>run rerun camCol field rowc colc</th>
<th>ra</th>
<th>dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1336 2 1 11 187.4 1635.3</td>
<td>251.2860</td>
<td>64.2985</td>
</tr>
<tr>
<td>1336 2 1 11 379.6 716.0</td>
<td>251.0971</td>
<td>64.2357</td>
</tr>
<tr>
<td>1336 2 1 11 542.9 327.7</td>
<td>251.0266</td>
<td>64.2010</td>
</tr>
<tr>
<td>1336 2 1 11 661.3 196.4</td>
<td>251.0098</td>
<td>64.1830</td>
</tr>
<tr>
<td>1336 2 1 11 873.6 811.6</td>
<td>251.1732</td>
<td>64.1913</td>
</tr>
<tr>
<td>1336 2 1 11 250.6 523.0</td>
<td>251.0389</td>
<td>64.2393</td>
</tr>
<tr>
<td>1336 2 1 11 261.0 1540.5</td>
<td>251.2723</td>
<td>64.2867</td>
</tr>
<tr>
<td>1336 2 1 11 324.7 653.6</td>
<td>251.0768</td>
<td>64.2382</td>
</tr>
<tr>
<td>1336 2 1 11 338.3 1052.6</td>
<td>251.1693</td>
<td>64.2558</td>
</tr>
<tr>
<td>1336 2 1 11 392.4 1749.8</td>
<td>251.3346</td>
<td>64.2836</td>
</tr>
</tbody>
</table>
Query both the Spectroscopy and Photometry

/*
 * Galaxy target selection with spectroscopic redshifts
 */

declare @database char set @database =
declare @pi float set @pi = 3.141592654
declare @BLENDED int set @BLENDED = dbo.fPhotoFlags('BLENDED')
declare @BRIGHT int set @BRIGHT = dbo.fPhotoFlags('BRIGHT')
declare @EDGE int set @EDGE = dbo.fPhotoFlags('EDGE')
declare @NODEBLEND int set @NODEBLEND = dbo.fPhotoFlags('NODEBLEND')
declare @SATURATED int set @SATURATED = dbo.fPhotoFlags('SATURATED')
declare @bad_flags int set @bad_flags = (@SATURATED | @BRIGHT | @EDGE)
declare @maglim float set @maglim = 17.77
declare @SBlim float set @SBlim = 24.5
declare @delta_psf_model float set @delta_psf_model = 0.3

select top 10
str(gal.ra,9,4) as ra, str(gal.dec,8,4) as dec,
'|' as '|', cast(spec.type as char (9)) as type,
str(spec.z,7,4) as Z,
dbo.fSpecZStatusN(spec.zStatus) as status,
dbo.fGetUrlSpecImg(spec.specObjID) as Spectra
from
from
  @database..PhotoPrimary as gal,
  @database..specObj as spec
where
gal.objID = spec.objID and
  -- Our star-galaxy separation and target selection
  psfMag_r - modelMag_r >= @delta_psf_model and
  petroMag_r - reddening_r <= @maglim and
  petroMag_r - 2.5*log10(2*@pi*petroR50_r*petroR50_r) < @SBlim and
  -- Check flags
  (flags & @bad_flags) = 0 and
  ((flags & @BLENDED) = 0) or ((flags & @NODEBLEND) != 0))
<table>
<thead>
<tr>
<th>ra</th>
<th>dec</th>
<th>type</th>
<th>Z</th>
<th>status</th>
<th>Spectra</th>
</tr>
</thead>
<tbody>
<tr>
<td>251.3749</td>
<td>64.1473</td>
<td>GALAXY</td>
<td>0.0689</td>
<td>XCORR_EMLINE</td>
<td>http:...</td>
</tr>
<tr>
<td>251.2649</td>
<td>64.0215</td>
<td>GALAXY</td>
<td>0.0677</td>
<td>XCORR_EMLINE</td>
<td>http:...</td>
</tr>
<tr>
<td>251.5386</td>
<td>63.8977</td>
<td>GALAXY</td>
<td>0.0695</td>
<td>XCORR_EMLINE</td>
<td>http:...</td>
</tr>
<tr>
<td>251.5323</td>
<td>63.7413</td>
<td>STAR_BHB</td>
<td>-0.0001</td>
<td>XCORR_EMLINE</td>
<td>http:...</td>
</tr>
<tr>
<td>251.6382</td>
<td>63.6581</td>
<td>GALAXY</td>
<td>0.1048</td>
<td>XCORR_HIC</td>
<td>http:...</td>
</tr>
<tr>
<td>252.0256</td>
<td>63.5788</td>
<td>GALAXY</td>
<td>0.0333</td>
<td>XCORR_HIC</td>
<td>http:...</td>
</tr>
<tr>
<td>251.8941</td>
<td>63.4230</td>
<td>GALAXY</td>
<td>0.0967</td>
<td>INCONSISTENT</td>
<td>http:...</td>
</tr>
<tr>
<td>252.3079</td>
<td>63.1463</td>
<td>GALAXY</td>
<td>0.1045</td>
<td>INCONSISTENT</td>
<td>http:...</td>
</tr>
<tr>
<td>252.6909</td>
<td>62.8880</td>
<td>GALAXY</td>
<td>0.0681</td>
<td>XCORR_HIC</td>
<td>http:...</td>
</tr>
<tr>
<td>252.4744</td>
<td>62.8420</td>
<td>GALAXY</td>
<td>0.0366</td>
<td>XCORR_EMLINE</td>
<td>http:...</td>
</tr>
</tbody>
</table>

There’s a problem with this query; it doesn’t return the objects which passed galaxy target selection but for which we have no spectrum.
There are Two Ways to write that Query

```
from @database..PhotoPrimary as gal,
     @database..specObj as spec
where gal.objID = spec.objID
   -- Our star-galaxy separation and target selection
```

and

```
from @database..PhotoPrimary as gal
join @database..specObj as spec
on gal.objID = spec.objID
where
   -- Our star-galaxy separation and target selection
```

The two are exactly equivalent; one uses `join ... on` whereas the other prefers to put the `on` condition into the `where` clause. We’re about to use a generalisation of this `join` syntax to include non-targetted galaxies in our outputs.
/*
 * Galaxy target selection including spectroscopic redshifts where available
 */
declare @database char set @database =
declare @pi float set @pi = 3.141592654
--
declare @BLENDED int set @BLENDED = dbo.fPhotoFlags('BLENDED')
declare @BRIGHT int set @BRIGHT = dbo.fPhotoFlags('BRIGHT')
declare @EDGE int set @EDGE = dbo.fPhotoFlags('EDGE')
declare @NODEBLEND int set @NODEBLEND = dbo.fPhotoFlags('NODEBLEND')
declare @SATURATED int set @SATURATED = dbo.fPhotoFlags('SATURATED')
--
declare @bad_flags int set @bad_flags = (@SATURATED | @BRIGHT | @EDGE)
--
declare @maglim float set @maglim = 17.77
declare @SBlim float set @SBlim = 24.5
declare @delta_psf_model float set @delta_psf_model = 0.3
--
select top 10
str(gal.ra,9,4) as ra, str(gal.dec,8,4) as dec,
'|' as '|' , cast(ISNULL(spec.type, 'NULL ') as char (9)) as type,
ISNULL(str(spec.z,7,4), 'NULL ') as Z,
ISNULL(dbo.fSpecZStatusN(spec.zStatus), 'NULL ') as status,
ISNULL(dbo.fGetUrlSpecImg(spec.specObjID), 'NULL ') as Spectra
--
from
from @database..PhotoPrimary as gal
left outer join @database..specObj as spec
on gal.objID = spec.objID
where
  -- Our star-galaxy separation and target selection
  psfMag_r - modelMag_r >= @delta_psf_model and
  petroMag_r - reddening_r <= @maglim and
  petroMag_r - 2.5*log10(2*pi*petroR50_r*petroR50_r) < @SBlim and
  -- Check flags
  (flags & @bad_flags) = 0 and
  (((flags & @BLENDED) = 0) or ((flags & @NODEBLEND) != 0))

Here the left outer join includes rows for which no spectrum is available, returning NULL. The select has to handle these NULLs.

The phrase
  ISNULL(str(spec.z,7,4), 'NULL')
isn't actually legal ANSI SQL (it's a SQL-server extension); the legal version is the rather wordier
  (case when spec.z is NULL then 'NULL' else str(spec.z,7,4) end)
<table>
<thead>
<tr>
<th>ra</th>
<th>dec</th>
<th>type</th>
<th>Z</th>
<th>status</th>
<th>Spectra</th>
</tr>
</thead>
<tbody>
<tr>
<td>251.2860</td>
<td>64.2985</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
<tr>
<td>251.0971</td>
<td>64.2357</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
<tr>
<td>251.3141</td>
<td>63.9971</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
<tr>
<td>251.3144</td>
<td>63.9881</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
<tr>
<td>251.3085</td>
<td>63.9892</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
<tr>
<td>251.5633</td>
<td>64.0398</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
<tr>
<td>251.3749</td>
<td>64.1473</td>
<td>GALAXY</td>
<td>0.0689</td>
<td>XCORR_EMLINE</td>
<td>http:...</td>
</tr>
<tr>
<td>251.2649</td>
<td>64.0215</td>
<td>GALAXY</td>
<td>0.0677</td>
<td>XCORR_EMLINE</td>
<td>http:...</td>
</tr>
<tr>
<td>251.4962</td>
<td>64.0124</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
<tr>
<td>251.3497</td>
<td>63.9756</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL NULL NULL</td>
</tr>
</tbody>
</table>
A Query that (Implicitly) Creates a Temporary Table

-- Make a temporary table

```
declare @database char  set @database = Robert
select *
from
  (select run, rerun,
       count(*) as nobj
  from @database..photoObj
  group by run, rerun
  ) as "tmpTable"
```
Make a temporary table (called `tmpTable`) and select its contents. This isn’t actually exactly a temporary table in the usual SQL-server sense, but I find it helpful to think of it that way.

The `count(*) as nobj` says return the total number of rows in the `@database..photoObj` table; the `group by run, rerun` says that the counting should be done separately for each pair of values (`run`, `rerun`).
Another version of “What’s Loaded”

-- What’s loaded in skyserver?

--

declare @database char set @database = Robert

select distinct
    str(seg.run, 4) as run, str(seg.rerun, 3) as rerun,
    str(field0,3) as field0, str(field0 + nfields - 1,4) as field1,
    str(nobj, 7) as nobj,
    photoVersion
from
    @database..Segment as seg,
    (select
        count (*) as nobj, run, rerun
    from
        @database..photoObj
    group by run, rerun
    ) as "fieldSummary"
where
    seg.run = fieldSummary.run and seg.rerun = fieldSummary.rerun
order by run
We’ve included the number of objects in each run in the output.
Fix the field1 Values Using Another Temporary Table

-- What’s loaded in skyserver?

declare @database char set @database = Robert

select distinct
    str(seg.run, 4) as run,
    str(seg.rerun, 3) as rerun,
    str(field0_rhl,3) as field0,
    str(field1_rhl,3) as field1,
    photoVersion
from @database..Segment as seg,
     (select
         min (field) as field0_rhl,
         max (field) as field1_rhl,
         run, rerun
     from @database..field
    group by run, rerun
     ) as "runSummary"
where seg.run = runSummary.run and seg.rerun = runSummary.rerun
order by run
run rerun field0 field1 photoVersion

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>745</td>
<td>672</td>
<td>395</td>
<td>514</td>
<td>v5_3_31</td>
<td></td>
</tr>
<tr>
<td>756</td>
<td>672</td>
<td>680</td>
<td>796</td>
<td>v5_3_31</td>
<td></td>
</tr>
<tr>
<td>1336</td>
<td>14</td>
<td>11</td>
<td>93</td>
<td>v5_3_31</td>
<td></td>
</tr>
<tr>
<td>1339</td>
<td>14</td>
<td>11</td>
<td>92</td>
<td>v5_3_31</td>
<td></td>
</tr>
</tbody>
</table>
Calculate the źhed point from the skyserver data

declare @database char set @database = Robert
declare @maglim int set @maglim = 18
declare @star int set @star = dbo.fPhotoType('star')
declare @bad_flags int set @bad_flags = (dbo.fPhotoFlags('SATURATED') | 
    dbo.fPhotoFlags('BRIGHT') | dbo.fPhotoFlags('EDGE'))

-- Possible restriction on runs to be processed
declare @select_run char set @select_run = -- run == 745 and

-- Our Query

select field.pspStatus,
blue.run, blue.camCol, blue.field, N,
gr as zhed_gr,
C0 + gr*C1 as zhed_ri,
C0, C1
from

@database..field as field,
(select -- Fit straight line
    run as run, camCol as camCol, field as field, fieldId, N,
    (-sum_x*sum_xy + sum_xx*sum_y)/(n*sum_xx - sum_x*sum_x) as C0,
    (N*sum_xy - sum_x*sum_y)/(n*sum_xx - sum_x*sum_x) as C1
from
    (select -- Blue part of locus
        run as run, camCol as camCol, field as field, fieldId,
        count (*) as N,
        sum (g - r) as sum_x,
        sum ((g - r)*(g - r)) as sum_xx,
        sum (r - i) as sum_y,
        sum ((g - r)*(r - i)) as sum_xy
from
    (select
        run, camCol, field, fieldId,
        (psfMag_g - reddening_g) as g,
        (psfMag_r - reddening_r) as r,
        (psfMag_i - reddening_i) as i
    from @database..PhotoPrimary
    where
        @select_run
        (flags & @bad_flags) = 0 and nchild = 0 and
        type = @star and
        -- Not too faint
        psfMag_i < @maglim
    ) as "obj1"
where
    -- Choose stars in nearly horizontal part of g-r-i diagram
    g - r between 0.3 and 1.1 and
    r - i between -0.1 and 0.6
group by run, camCol, field, fieldId
) as "blue",
) as "blue",

(select -- Red part of locus
    fieldId,
    avg(g - r) as gr-- I'd prefer the median
from
    (select
        fieldId,
        (psfMag_g - reddening_g) as g,
        (psfMag_r - reddening_r) as r,
        (psfMag_i - reddening_i) as i
        from @database..PhotoPrimary
    where
        @select_run
        (flags & @bad_flags) = 0 and nchild = 0 and
        type = @star and
        -- Not too faint
        psfMag_i < @maglim
    ) as "obj2"
where
    -- Choose stars in vertical part of g-r-i diagram
    g - r between 1.1 and 1.6 and
    r - i between 0.8 and 1.4
    group by fieldId
) as "red"
where
    blue.fieldId = red.fieldId and
    blue.fieldId = field.fieldId
order by blue.run, blue.camCol, blue.field
But what if I want an average for each run/camCol? Ask and ye shall receive...

```
pspStatus run camCol field N zhed_gr zhed_ri C0 C1
0 745 1 395 22 1.364067 0.549448 0.003595 0.400166
0 745 1 396 33 1.381796 0.469429 0.001625 0.338548
0 745 1 397 40 1.329576 0.499432 -0.009701 0.382928
0 745 1 398 37 1.434749 0.538586 0.003014 0.373286
0 745 1 399 26 1.380334 0.509576 -0.023451 0.386158
0 745 1 400 36 1.361475 0.513623 -0.011265 0.385528
...```
-- Calculate the zhed point from the skyserver data
--
declare @database char    set @database = -- Robert
declare @Nmin int         set @Nmin = 0 -- Minimum number of stars/field
declare @maglim int       set @maglim = 18
declare @star int         set @star = dbo.fPhotoType('star')
declare @bad_flags int    set @bad_flags = (dbo.fPhotoFlags('SATURATED') | db 
| dbo.fPhotoFlags('EDGE'))
declare @good char        set @good = (1=1 or field.pspStatus = 0) and blue.N > @Nmin
-- Possible restriction on runs to be processed
declare @select_run char  set @select_run = -- run = 1339 and camCol = 1 and
-- Our Query
--
select blue.run, blue.camCol,
    sum (case when @good then blue.N else 0 end ) as N,
avg(case when @good then gr else NULL end ) as zhed_gr,
avg(case when @good then C0 + gr*C1 else NULL end ) as zhed_ri
from
@database..field as field,
(select -- Fit straight line
    run as run, camCol as camCol, field as field, fieldId, N,
    (-sum x*sum xy + sum xx*sum y)/(n*sum xx - sum x*sum x) as C0,
    (N*sum xy - sum x*sum y)/(n*sum xx - sum x*sum x) as C1
from
(select -- Blue part of locus
    run as run, camCol as camCol, field as field, fieldId,
    count (*) as N,
    sum (g - r) as sum _x,
    sum ((g - r)*(g - r)) as sum _xx,
    sum (r - i) as sum _y,
    sum ((g - r)*(r - i)) as sum _xy
from
(select
    run, camCol, field, fieldId,
    (psfMag.g - reddening_g) as g,
    (psfMag.r - reddening_r) as r,
    (psfMag.i - reddening_i) as i
from @database..PhotoPrimary
where
    @select _run
    (flags & @bad_flags) = 0 and nchild = 0 and
    type = @star and
    -- Not too faint
    psfMag_i < @maglim
) as "obj1"
where
    -- Choose stars in nearly horizontal part of g-r-i diagram
    g - r between 0.3 and 1.1 and
    r - i between -0.1 and 0.6
group by run, camCol, field, fieldId
) as "blue"
(select -- Red part of locus
    fieldId,
    avg(g - r) as gr-- I’d prefer the median
from
  (select
    fieldId,
    (psfMag.g - reddening.g) as g,
    (psfMag.r - reddening.r) as r,
    (psfMag.i - reddening.i) as i
  from @database..PhotoPrimary
  where
    @select_run
      (flags & @bad_flags) = 0 and nchild = 0 and
      type = @star and
      -- Not too faint
      psfMag.i < @maglim
  ) as "obj2"
where
  -- Choose stars in vertical part of g-r-i diagram
  g - r between 1.1 and 1.6 and
  r - i between 0.8 and 1.4
  group by fieldId
) as "red"
where
  blue.fieldId = red.fieldId and
  blue.fieldId = field.fieldId
group by blue.run, blue.camCol
order by blue.run, blue.camCol
The results of the žhed query. The white cross is the canonical cosmic žhed point, with $±1\%$ error bars.