iDR4: Example Queries for the GES archive

This document pertains to release iDR2iDR3 of the GES archive. It should not be used with earlier releases of the archive; some aspects of the discussion and examples are incorrect for previous releases.

This document presents a number of examples of typical queries of the GES archive. The examples are intended to be similar to the queries which users new to the archive will often make.

Prerequisites

The GES archive is searched using queries expressed in the Structured Query Language (SQL), which is the standard way of manipulating relational databases. The examples presented here assume familiarity with both the rudiments of SQL and use of the GES SQL query pages.

A beginner's introduction to SQL is available here.

Instructions for using the GES SQL query pages are available here.

The examples are divided into a number of sections based on the type of query being performed:

Queries on Target Stars

These queries return lists of target stars selected from the archive. Note that they do *not* return lists of spectra of those stars: a given star may be observed more than once. Nor do they return lists of recommended (or other) physical parameters (effective temperature *etc.*) and abundances. Subsequent sections give examples of querying these quantities.

Aside on star names

The naming of stars, in catalogues and elsewhere, can be surprisingly problematic, with the brighter stars, in particular, enjoying a variety of designations. Some catalogue naming schemes derive a star's name from its celestial coordinates, others do not; there are pros and cons to both approaches.

In the Gaia-ESO Survey (GES) survey each star is assigned a name based on its celestial coordinates. This name is formed by concatenating the sexagesimal Right Ascension in hours with the sexagesimal Declination in degrees according to the following scheme:

```
hhmmssss±ddmmsss
```

where ± specifies the sign of the Declination, either '+' or '-'. Several of the tables in the archive include the name of the target stars in this format, in a column called 'cName'. The names tabulated in column 'cName' are derived from each star's equatorial coordinates as supplied by ESO. Currently the following tables contain a 'cName' column:

```
Target
Spectrum
AstroAnalysis
```

1. **Is a given star in the archive?**

   A list of all the stars in the GES archive is stored in the 'Target' table. This table contains two columns of star names. One is column 'cName', as described above, which contains the name

derived from the star's equatorial coordinates. The other column is called 'esoName' and contains the star's name as returned by ESO (and originally specified by the GES Consortium when the star was selected for inclusion in the survey), in the same format as 'cName'. Usually the two names will be the same for a given star, but they may sometimes differ in the least significant digit in either coordinate. The following queries select all the information in table 'Target' for star 11053303-7700120 using the 'cName' and 'esoName' column respectively (for this star the two names are the same):

```
SELECT * FROM Target WHERE cName ='11053303-7700120';

SELECT * FROM Target WHERE esoName='11053303-7700120';
```

2. **Which (if any) of the following list of stars are in the archive?**

If you wish to check which of the stars in a list are in the archive then the SQL 'IN' clause can be used. The following queries show the use of this clause for selections on both the 'cName' and 'esoName' columns. Note that in principle there is no restriction on the size of the list. Also note that only stars in the list that are found in the archive will be listed; any which are not in the archive will not appear (star 'bad-name' is included in the example to illustrate this point).

```
SELECT * FROM Target WHERE cName IN('11034945-7700101', '11044460-7706240',
'bad-name', '11053303-7700120');

SELECT * FROM Target WHERE esoName IN('11034945-7700101', '11044460-7706240',
'bad-name', '11053303-7700120');
```

Any stars not in your list can be identified using a query of the (slightly cumbersome) form:

```
SELECT * FROM ( VALUES ('11034945-7700101'), ('11044460-7706240'), ('bad-
name'), ('11053303-7700120')) AS starList(starName) WHERE starName NOT IN
(SELECT cName FROM Target);
```

3. **Select all the stars within a given range of RA and Dec.**

The J2000 Right Ascension and Declination of stars in the GES survey are respectively stored in columns 'ra' and 'dec' of table 'Target'. Both coordinates are stored in decimal degrees (note in particular that the Right Ascension is *not* stored in hours) and southern Declinations are negative. Thus, the first step is to convert the coordinates of your range to this format. The stars enclosed in the region can then be found with a query of the form:

```
SELECT * FROM Target WHERE (ra BETWEEN ra_min AND ra_max) AND (dec BETWEEN
dec_min AND dec_max);
```

Suppose your range was 70 to 80 degrees of Right Ascension and -45 to -30 degrees of Declination then the corresponding SQL would be:

```
SELECT * FROM Target WHERE (ra BETWEEN 70 AND 80) AND (dec BETWEEN -45 AND
-30);
```

4. **List the magnitudes and colours available for a given set of stars**

Magnitudes and colours in table Target

Table Target includes magnitudes and colours for the stars in the GES survey that have been obtained from a number of public surveys. The surveys included are as follows:

Survey	Acronym	Photometric bands included
VISTA VHS	-	j, h, k
Two-micron All Sky Survey	2MASS	j, h, k
UK Infrared Deep-Sky Surveys	UKIDSS	j, h, k
Sloan Digital Sky Survey	SDSS	u, g, r, i, z
AAVSO Photometric All-Sky Survey	APASS	v, b, g, r, i
Johnson magnitudes	-	u, b, v, r, i

Each photometric band or magnitude for each survey is is included as a column of Target and the name of this column is formed in a uniform way as follows:

```
band name + 'Mag' + survey acronym
```

In addition Target includes:

- for each magnitude an estimate of its error (with a column name formed by appending 'Err' to the name of the corresponding magnitude column),
- for every survey a measure of the angular separation, in seconds of arc, between the GES coordinates for the star and its survey coordinates (with a column named formed by inserting the string 'dist' before the survey's acronym).

These rules have the following exceptions:

- The VISTA VHS survey and the Johnson magnitudes have no acronym.
- The Johnson magnitudes do not have corresponding error columns or an angular separation column.

Thus, for each of the surveys the names of the magnitude and error columns for the first photometric band and the angular separation column are as follows:

Survey	magnitude	error	angular separation
VISTA VHS	jMag	jMagErr	distVISTA
2MASS	jMag2mass	jMag2massErr	dist2mass
UKIDSS	jMagUkidss	jMagUkidssErr	distUkidss
SDSS	uMagSdss	uMagSdssErr	distSdss
APASS	vMagApass	vMagApassErr	distApass
Johnson catalogue magnitudes	uMag	-	-

To list the VISTA VHS magnitudes and errors and the angular separation for a set of stars:

```
SELECT
  cname, jMag, jMagErr, hMag, hMagErr, ksMag, ksMagErr, distVISTA
FROM
  Target
WHERE
  cname IN('00024679-4702544', '00315212-4402080', '11034945-7700101', '23570212-4803198')
ORDER BY cname;
```

Magnitudes are not available for every star in every band of every survey. Where no measurement is available the usual approach of storing a default value well outside the range of plausible values for a stellar magnitude is adopted. In the example ksMag and its error are not available for star 00315212-4402080 and no VISTA VHS magnitudes are available for 11034945-7700101.

Queries on Spectra

These queries return details of spectra available in the GES archive. Recall that each star may have more than one spectrum (because it was observed more than once).

1. **Which (if any) spectra are available for a given star?**

   All the spectra in the GES archive are listed in table 'Spectrum'. This table contains auxiliary information about each spectrum (metadata in the jargon of computer science), such as a magnitude estimate for the target star, the observing priority, the exposure time, *etc*. For convenience of identifying stars a 'cName' column is included listing each star's name. Other information pertaining to the target star, rather than to the spectrum, can be listed by joining the 'Spectrum' and 'Target' tables. The spectra themselves are not stored in a database table but as FITS files. As a first example, to list all the spectra for star 11053303-7700120:

   ```
   SELECT * FROM Spectrum WHERE cName ='11053303-7700120';
   ```

2. **Which (if any) spectra are available for a given list of stars?**

   Queries to list all the spectra available for a list of stars are similar to those for a single star (above) but use the 'IN' clause to specify the star list. Note that in principle there is no restriction on the size of the star list. Also note that only stars in the list that are found in the archive will be listed; any which are not in the archive will not appear (star 'bad-name' is included in the example to illustrate this point). The following query lists all the spectra available for a list of stars:

   ```
   SELECT * FROM Spectrum WHERE cName IN('11034945-7700101', '11044460-7706240',
   ```

```
'bad-name', '11053303-7700120');
```

Any stars with no spectra can be identified using a query of the (slightly cumbersome) form:

```
SELECT * FROM ( VALUES ('11034945-7700101'), ('11044460-7706240'), ('bad-
name'), ('11053303-7700120')) AS starList(starName) WHERE starName NOT IN
(SELECT cName FROM Spectrum);
```

3. **How many spectra are available for each of a given list of stars?**

   The following query counts the number of spectra for each of the stars in a list. Note that any stars for which there are no spectra are not included in the returned list (star 'bad-name' is included in the example to illustrate this point).

   ```
   SELECT cName AS 'star', COUNT(cName) AS 'no. of spectra' FROM Spectrum WHERE
   cName IN ('11034945-7700101', '11044460-7706240', 'bad-name',
   '11053303-7700120') GROUP BY cName ORDER BY cName;
   ```

   See the previous example for how to check for any stars in the list which have no spectra.

4. **List all the spectra available for all the stars within a given range of RA and Dec.**

   As previously described queries to find all the stars in a given region of Right Ascension and Declination must be made on the 'Target' table because it is the only table to list each star's celestial coordinates. The 'Target' table can then be joined with the 'Spectrum' table to find the spectra available for each star. For example, to list the name, equatorial coordinates, exposure time, signal-to-noise ratio and magnitude of all the spectra for all the stars in the range 70 to 80 degrees of Right Ascension and -45 to -30 degrees of Declination:

   ```
   SELECT
     tg.cname, tg.ra, tg.dec, sp.expTime, sp.snr, sp.mag
   FROM
     Target tg,
     Spectrum sp
   WHERE tg.targetId = sp.targetId
     AND (tg.ra BETWEEN 70 AND 80)
     AND (tg.dec BETWEEN -45 AND -30);
   ```

   Internal identifiers in GES archive tables

   The above query joined the 'Target' and 'Spectrum' tables. This operation could have been done using the 'cName' column to identify corresponding rows in the tables since this column occurs in both tables. However, it was actually done using a column called 'targetId' which also occurs in both tables. These 'targetId' columns have a similar purpose to 'cName': to uniquely identify a given star (or 'target') and hence to allow joins to proceed by identifying rows in the two tables pertaining to the same star.

   The advantages of using column 'targetId' rather than column 'cName' are that the former is numeric and indexed, which permits fast and efficient access. Consequently 'targetId' should be used in preference to 'cName' to identify corresponding stars in two or more tables. However, you should never need to know the actual values of 'targetId' (they are just numbers) as they are invented internally for the archive and have no wider astronomical significance. Moreover, they are created afresh for each release of the database, so *do not* rely on them remaining the same in different releases. For example, if you make a note of the 'targetId' of your favourite star in the current release and then query the following release, when it appears, using this value you will retrieve data for a different star.

   The archive contains several other identifiers in addition to 'targetId'. For example, every row in the 'Spectrum' table also has an identifier, column 'specId', which uniquely identifies it. This identifier is used, for example, in identifying the spectrum from which individual astrophysical analysis parameters (effective temperature *etc.*) have been derived. Several different types of identifier will appear in the examples below.

5. **Which (if any) spectra are available for a given list of stars and with which spectrograph were they obtained?**

   The 'Spectrum' table does not include a column listing the spectrograph with which it was acquired, so a more involved approach is required to show the instrument used.

   GES spectra are all obtained using one of two multi-object spectrographs: GIRAFFE and UVES. Such instruments acquire a group (or 'frame') of spectra simultaneously (and are related to ESO's 'OBs' or 'observing blocks'). The 'SpecFrame' table lists all such frames included in the GES

survey and contains details pertaining to the entire frame rather than individual spectra. It includes column 'instrument' which lists the spectrograph used. The values corresponding to the two instruments are: 'GIRAFFE' and 'UVES' (note that the values are in upper case). To show which spectrograph was used, the rows selected from the 'Spectrum' table must be joined to the rows for their parent frames in the 'SpecFrame' table. The 'SpecFrame' table has a column, 'specFrameId', which contains an integer value that uniquely identifies each specFrame (see the note about identifiers, above). Table 'Spectrum' also contains a 'specFrameId' column and it lists the identifier of the specFrame from which each spectrum was extracted. Thus:

```
SELECT
  spec.cName, frame.instrument
FROM
  Spectrum spec,
  SpecFrame frame
WHERE spec.specFrameId = frame.specFrameId
  AND spec.cName IN ('11034945-7700101', '11044460-7706240', '21101955-0200414');
```

A similar query can be used to show only the spectra obtained with a given spectrograph:

```
SELECT
  spec.cName, frame.instrument
FROM
  Spectrum spec,
  SpecFrame frame
WHERE spec.specFrameId = frame.specFrameId
  AND spec.cName IN ('11034945-7700101', '11044460-7706240', '21101955-0200414')
  AND frame.instrument = 'UVES';
```

It is also possible to combine the 'Spectrum', 'SpecFrame' and 'Target' tables to list all the spectra available in a given range of Right Ascension and Declination and to show the spectrograph with which they were observed:

```
SELECT
  tg.cname, tg.ra, tg.dec, frame.instrument, sp.expTime
FROM
  Target tg,
  Spectrum sp,
  SpecFrame frame
WHERE tg.targetId = sp.targetId
  AND sp.specFrameId = frame.specFrameId
  AND (tg.ra BETWEEN 70 AND 80)
  AND (tg.dec BETWEEN -45 AND -30);
```

The values of column 'instrument' in the 'SpecFrame' table corresponding to the two spectrographs used are 'GIRAFFE' and 'UVES', however to check which values actually occur in the table:

```
SELECT DISTINCT instrument FROM SpecFrame;
```

or to check the number of times each occurs and to order the resulting list alphabetically by instrument:

```
SELECT instrument, COUNT(instrument)
FROM   SpecFrame
GROUP BY instrument
ORDER BY instrument;
```

You are likely to find that your queries of spectra will often also involve columns from 'SpecFrame', as in the above examples: some of the properties that are naturally thought of as attributes of a spectrum (such as the instrument used to acquire it or its exposure time) are stored in 'SpecFrame' because they are common to all the spectra in the frame. For convenience all the columns in table 'Spectrum' and some of the more often-used columns in 'SpecFrame' are combined in the *view* 'SpectrumAndFrame'. In the context of relational databases a view is simply a pre-defined combination or subset of tables provided along with the basic tables of the database. The views included with the GES archive work exactly like tables and are queried in the same way. The 'SpecFrame' column 'instrument' is included in the 'SpectrumAndFrame' view, so, the previous example to list the instrument used to acquire each of a list of spectra could be more conveniently and concisely expressed as:

```
SELECT
  cName, instrument
FROM
  SpectrumAndFrame
WHERE cName IN ('11034945-7700101', '11044460-7706240', '21101955-0200414');
```

Often you will find it more convenient to use view 'SpectrumAndFrame' than table 'Spectrum'.

6. **How do I list all the details available for a given Spectrum Frame?**

Details available for each Spectrum Frame

Every spectrum frame in the archive was ingested into it from a FITS file, and the columns of table SpecFrame, such as telescope, instrument or grating, come from keywords in this FITS file. The spectrum frame files contain many keywords, but only a few are imported into columns in table SpecFrame; the ones most likely to be useful when querying spectra. However, all the keywords read from the FITS file are available in the archive; they are stored in table SpecFrameFitsKey.

Each keyword is not stored as a separate column in SpecFrameFitsKey because there are too many keywords for this to be practical. Rather each keyword is stored as a name / value pair with the name of the keyword stored in column 'name' and the associated value in column 'value'. Finally, just as column specFrameID in table Spectrum allows spectra to be associated with their parent spectrum frame (as discussed above) so a similar specFrameID column is included in table SpecFrameFitsKey to allow FITS keywords to be associated with their parent spectrum frame.

First, to determine the number of keywords available for a given spectrum frame simply query table SpecFrameFitsKey specifying the specFrameID of the frame:

```
SELECT
  COUNT(*)
FROM
  SpecFrameFitsKey
WHERE specFrameID = 8;
```

and similarly, to list all the keywords for this frame:

```
SELECT
  specFrameID, extNum, name, value, units
FROM
  SpecFrameFitsKey
WHERE specFrameID = 8;
```

There are a number of points to note from these examples:

1. Every spectrum frame has between 1000 and 3000 keywords associated with it; the example had 1178, which is typical.
2. In addition to columns for its name and value each keyword also has columns tabulating any units the value may have and any description provided for it.
3. The original FITS files contained multiple FITS extensions, each with its own set of keywords. Keyword names are often duplicated between extensions; indeed many of the extensions include copies of keywords from the first (or primary) extension. Thus, table SpecFrameFitsKey includes column extNum which tabulates the extension number that the keyword came from; you may need to use it to identify a particular keyword.
4. The keyword names conform to the conventions for ESO Hierarchical FITS rather than those for standard FITS keywords (if you care about such things).

Listing one or more named keywords for a frame is more likely to be useful than listing all the frame's keywords because of the large number available. Remember to either specify the appropriate extension number in your selection or to list the extension number amongst the columns you select. Often simply selecting keywords from the first (or primary) extension will be adequate:

```
SELECT
  specFrameID, extNum, name, value, units
FROM
```

```
  SpecFrameFitsKey
WHERE specFrameID = 8
  AND extNum = 0
  AND name IN ('WAVELMAX', 'WAVELMIN', 'SPEC_BIN', 'SPEC_BW', 'SPEC_RES', 'SPEC_VAL')
ORDER BY name;
```

Often it will be convenient to join table SpecFrameFitsKey to tables SpecFrame or Spectrum so that you do not need to specify individual specFrameIDs. For example, to see the values of keywords that apply to each of the spectra that have been obtained for a given star:

```
SELECT
  spec.cname, keys.extNum, keys.name, keys.value, keys.units,
  spec.specFrameID, keys.specFrameID
FROM
  Spectrum spec,
  SpecFrameFitsKey keys
WHERE spec.cname = '11034945-7700101'
  AND keys.extNum = 0
  AND keys.name IN ('WAVELMAX', 'WAVELMIN', 'SPEC_BIN', 'SPEC_BW', 'SPEC_RES', 'SPEC_VAL')
  AND spec.specFrameID = keys.specFrameID
ORDER BY spec.specFrameID, name;
```

Columns from table SpecFrame can similarly be included:

```
SELECT
  spec.cname, frame.instrument, frame.grating,
  keys.extNum, keys.name, keys.value, keys.units,
  spec.specFrameID, frame.specFrameID, keys.specFrameID
FROM
  Spectrum spec,
  SpecFrame frame,
  SpecFrameFitsKey keys
WHERE spec.cname = '11034945-7700101'
  AND keys.extNum = 0
  AND keys.name IN ('WAVELMAX', 'WAVELMIN', 'SPEC_BIN', 'SPEC_BW', 'SPEC_RES', 'SPEC_VAL')
  AND spec.specFrameID = frame.specFrameID
  AND spec.specFrameID = keys.specFrameID
ORDER BY spec.specFrameID, name;
```

7. **How do I download a single spectrum (singleSpec file) from the archive?**

Spectrum files in the GES archive

The GES archive includes two types of files containing spectra: so-called **singleSpec** and **manySpec** files. In order to understand the difference recall that the GES observations are made with multi-object spectrographs which typically acquire a number of spectra simultaneously.

SingleSpec file

> contains a single one-dimensional spectrum of a single object. All the observations of the object made with a given instrument configuration have been co-added or stacked into a single spectrum.

ManySpec file

> contains a collection or frame of spectra acquired simultaneously. In the final manyspec files currently available all the spectra in the frame have been stacked from all the observations of the frame made with a given instrument configuration. The spectra themselves are one-dimensional but they are stored in a two-dimensional array (with one axis corresponding to wavelength and the other to an index for each each spectrum in the frame).

Note that in release iDR4, unlike earlier releases, manySpec files are only available for GIRAFFE spectra. Though there are circumstances where you may want either file (if both are available), usually singleSpec files will be more useful and convenient. The names of individual files in the sets are stored in two different tables in the archive, in both cases in a column called 'fileName':

Type of file	Table	File tabulated in column 'fileName'
SingleSpec	SpectrumGroup	FITS file of an individual stacked spectrum
ManySpec	SpecFrame	FITS file of a complete spectrum frame

Both singleSpec and manySpec files are stored in FITS format. A report describing the format is available for download (175 Kbyte).

Once you have identified a set of spectra that are of interest you can download singleSpec files containing them. Recall that each row in table 'Spectrum' corresponds to a single spectrum, identified by a unique 'specID', and tabulates the details characterising it. These details were extracted from keywords in the FITS files containing the spectrum and comprise the metadata available for it. For most spectra tabulated in 'Spectrum' a singleSpec file containing the spectrum is also available. The names of these files are tabulated in column 'fileName' of table 'SpectrumGroup'. If this column is included in a selection then an additional column, 'getFLink', is also listed which provides a link to the singleSpec file. Simply click on this link to retrieve a copy. Since column 'fileName' is in table 'SpectrumGroup' usually you will need to join it to table 'Spectrum' using the 'specID' column in both these tables. For example to download the spectra available for a given star:

```
SELECT DISTINCT
  spec.cName, spec.specID, spg.specID, spg.fileName
FROM
  Spectrum spec,
  SpectrumGroup spg
WHERE spec.specID = spg.specID
  AND cName ='11053303-7700120';
```

(The DISTINCT clause is simply to remove duplicate entries found in the 'SpectrumGroup' table; see the following section, Queries on Individual Analyses, for further discussion of 'SpectrumGroup'.) To retrieve the required files simply click on the appropriate entries in the 'getFLink' column in the listing generated by the query.

Clicking on individual links is impractical if you wish to download a large number of files. In this case see the item ' How do I download a large number of files?' below.

8. **How do I download a spectrum frame (manySpec) file from the archive?**

Recall that in release iDR4 manySpec files are only available for GIRAFFE spectra. The names of the manySpec files of frames available in the archive are tabulated in column 'fileName' of table 'SpecFrame' ('SpecFrame' basically contains FITS keywords extracted from these files). In a similar way to the singleSpec 'fileName' column in table the 'SpectrumGroup' columns in the previous example, if the 'specFrame' column 'fileName' is included in a selection then an additional column, 'getFLink', will be listed containing a link to download a copy of the manySpec file. For example, to download the spectrum frame manySpec file from which a spectrum of a given star was extracted then table 'SpecFrame' should be joined with table 'Spectrum':

```
SELECT
  spec.cName, spec.specFrameID, frame.specFrameID, frame.fileName, frame.instrument
FROM
  Spectrum spec,
  SpecFrame frame
WHERE spec.specFrameID = frame.specFrameID
  AND cName ='10442506-6415397';
```

To retrieve the required manySpec files simply click on the appropriate entries in the 'getFLink' column in the listing generated by the query, as in the example for spectrum singleSpec files (above).

As for the singleSpec files, clicking on individual links is impractical if you wish to download a large number of files. In this case again see the item 'How do I download a large number of files?' below.

9. **How do I download a large number of files from the archive?**

Clicking links to download individual files is fine to retrieve copies of a few files but impractical for a large number. An alternative mechanism is provided to handle this latter case. It uses common Unix shell commands and hence can be scripted. Two shell commands for downloading Web pages and files are in common use: `wget` and `curl`. `wget` has been around for a while whereas `curl` is newer. Either or both are likely to be available on Linux systems whereas recent versions of Mac OS X, at least, only have `curl`.

The function 'dbo.fWgetCmd' has been added to the SQL parser in the GES archive. It is given a single argument, the name of a column containing a file name, and it returns the appropriate `wget`

command to download a copy of the file. 'dbo.fWgetCmd' works with both the 'SpectrumGroup' (for singleSpec files) and 'SpecFrame' (for manySpec files) tables. Other columns can also be listed in the query. For example to select singleSpec files from 'SpectrumGroup':

```
SELECT DISTINCT
  spec.cName, spg.fileName, dbo.fWgetCmd(spg.filename) AS 'Download_command'
FROM
  Spectrum spec,
  SpectrumGroup spg
WHERE spec.specID = spg.specID
  AND cName ='11053303-7700120';
```

In this example only two rows are returned; in practice you would use 'dbo.fWgetCmd' in queries where you wished to select a larger number of spectra. Similarly, to select manySpec files from 'SpecFrame':

```
SELECT
  spec.cName, frame.fileName, dbo.fWgetCmd(frame.filename) AS 'Download_command'
FROM
  Spectrum spec,
  SpecFrame frame
WHERE spec.specFrameID = frame.specFrameID
  AND cName ='11053303-7700120';
```

Once your query has run the `wget` commands listed by 'dbo.fWgetCmd' can be executed as you prefer. However, shell script `crtdownloadscrpt.sh` is available to simplify this task. To use it proceed as follows.

1. Before submitting your query choose the 'ASCII FILE' option for the results file, so that the table generated will be written as a CSV file.
2. Retrieve a copy of `crtdownloadscrpt.sh` if you have not already done so (click here to retrieve a copy).
3. The retrieved file must be executable. Change its permission if necessary:

   ```
   chmod u+x ./crtdownloadscrpt.sh
   ```

4. `crtdownloadscrpt.sh` will read your CSV file of results and create a script to automatically download all the file URLs tabulated by 'dbo.fWgetCmd'. Any other columns tabulated in the CSV file will be ignored. `crtdownloadscrpt.sh` creates a script to retrieve the files using either `wget` or `curl`. In the case of `wget` it just extracts the entries generated by 'dbo.fWgetCmd'. For `curl` it substitutes `wget` with `curl` and makes the necessary changes to the command-line arguments.
5. To use `crtdownloadscrpt.sh` to generate a `wget` script start a Unix terminal window, make the directory containing the CSV file and `crtdownloadscrpt.sh` the current directory and type:

   `./crtdownloadscrpt.sh` *csv-file-name*

   or to generate a `curl` download script:

   `./crtdownloadscrpt.sh` *csv-file-name curl*

   (Any second argument to `crtdownloadscrpt.sh` will cause it to create a `curl` rather than a `wget` script.) The download script will have the same name as the CSV file but file type .sh.
6. Again you might need to change the permission of the new download script:

   `chmod u+x ./`*download-script*

7. Finally, run the new download script:

   `./`*download-script*

Queries on Nightly Spectra

Please note that in conformance with the normal Consortium policy you must obtain permission from the PIs before using the nightly spectra in publications or calculations leading to publications.

Types of spectra in the GES archive

The GES data archive contains only fully reduced wavelength and intensity calibrated spectra; it does not contain raw or partly reduced observations. Nonetheless it includes two types of spectra: 'stacked' and 'nightly' spectra:

Stacked Spectra

are typically the result of combining spectra taken over multiple nights and are necessarily composite, time-averaged spectra of the star observed. The details of stacked spectra are listed in table Spectrum. All the examples of querying spectra given hitherto in this tutorial have referred to stacked spectra.

Nightly spectra

contain the spectrum of a star as observed during a single night. Typically several nightly spectra will be combined to yield a single composite stacked spectrum.

Stacked spectra are entirely adequate for many sorts of astronomical investigation and have the advantage of being less noisy than nightly spectra because they are the result of combining more observations. You would use stacked spectra in preference to nightly ones unless there was a reason not to. However, for studies of variable or binary stars where some aspect of the spectrum changes with time you are likely to need to refer to the nightly spectra.

The details of the nightly spectra are tabulated in table SpectrumNightly which has columns somewhat similar to those of table Spectrum. Each nightly spectrum is identified by a unique sequence number or identifier, which is stored in column specNightlyID. Table SpectrumNightly also includes a column specID which lists the identifier of any stacked spectrum to which the nightly spectrum contributes. This column can be used to used to join tables SpectrumNightly and Spectrum, and thence join other tables such as SpecFrame or Target.

1. **List all the nightly spectra that constitute a given stacked spectrum**

   To list all the nightly spectra used to create a given stacked spectrum simply select all the entries in table SpectrumNightly with the spectrum identifier (column specID) of the stacked spectrum:

```
SELECT
  specNightlyID, specID, expTime, dateObs, mjdObs, airmassStart, rmsTrace
FROM
  SpectrumNightly
WHERE
  specID = '13918';
```

2. **List all the nightly spectra available for a given star**

   Similarly it is possible to list all the nightly spectra available for a given star, identified by its cName (recall that the archive can contain multiple stacked spectra for a given star). SpectrumNightly does not tabulate cName so it must be joined with Spectrum:

```
SELECT
  spec.cname, spec.specID, night.specNightlyID, night.expTime, night.dateObs, night.mjdObs
FROM
  SpectrumNightly night,
  Spectrum spec
WHERE night.specID = spec.specID
  AND spec.cname = '11010007-7738516'
ORDER BY dateObs;
```

   Note that in this example the nightly spectra are listed in order of the date on which they were observed which is likely to be useful when studying time-dependent phenomena.

3. **List all the radial velocities available for a given star**

   In studies of binary and multiple systems the radial velocities tabulated in table SpectrumNightly are likely to be of interest. SpectrumNightly contains two sets of radial velocity columns, one listing values computed from Giraffe spectra and the other values from UVES spectra. The details are:

Column	Description

vel	The radial velocity measured from Giraffe spectra.
velErr	The error on vel.
rv	The radial velocity measured from UVES spectra.
rvErr	The error on rv.

In both cases the radial velocities are heliocentric, have the usual convention that recessional velocities are positive and have been determined by cross-correlation. To see all the radial velocities available columns vel and rv must both be listed:

```
SELECT
  spec.cname, spec.specID, night.specNightlyID, night.dateObs,
  night.rv, night.rvErr, night.vel, night.velErr
FROM
  SpectrumNightly night,
  Spectrum spec
WHERE night.specID = spec.specID
  AND spec.cname = '11010007-7738516'
ORDER BY dateObs;
```

To see additional details, such as the instrument and grating setting used when the nightly spectra were observed, tables SpectrumNightly and Spectrum can be joined with table SpecFrame:

```
SELECT
  spec.cname, spec.specID, night.specNightlyID, night.dateObs,
  night.rv, night.rvErr, night.vel, night.velErr,
  frame.instrument, frame.grating
FROM
  SpectrumNightly night,
  Spectrum spec,
  SpecFrame frame
WHERE night.specID = spec.specID
  AND spec.specFrameId = frame.specFrameId
  AND spec.cname = '11010007-7738516'
ORDER BY dateObs;
```

Queries on Recommended Analyses

Aside on recommended and other parameters

This discussion applies to release GESiDR2; subsequent releases may be different and previous releases were different.

The spectra acquired as part of GES are reduced and analysed by the GES Consortium. The structure of the Consortium includes several **working groups** and each working group typically comprises several **nodes**.

Working groups

> typically coordinate analyses of a set of stars, usually of a given spectral class (or range of spectral classes) and/or observed with a given spectrograph (GIRAFFE or UVES). The actual analyses are performed by the nodes comprising the woking group.

Nodes

> within a working group perform an individual analysis on the stars assigned to the working group or a subset of them. The nodes use a variety of analysis methods to arrive at their sets of parameters and abundances.

Every node analysis produces a set of astrophysical parameters pertaining to the entire spectrum (such as the effective temperature or surface gravity) and, in some cases, abundances of a set of species for each star that it works on. The working group then combines these results into a single preferred set of astrophysical parameters and abundances that it recommends. The archive contains both these working group recommended parameters and the individual parameters computed by the various nodes. Moreover, often the individual nodes will perform two types of analyses: one where they determine both astrophysical parameters and abundances and a second in which they adopt the astrophysical parameters recommended by their parent working group and determine just the abundances.

Finally there is one working group (WG15) which does not perform analyses itself but produces a recommended, homogenised set of results from the recommended results of Working Groups 10-14 (the

selection criteria for recommended values are described in the WG15 report). *It is Consortium policy that the recommended, homogenised WG15 parameters and abundances must be used in all GES publications and calculations leading to such publications.*

All the different types of result are stored in table 'AstroAnalysis' and a set of views are defined on this table corresponding to each of the types of result (the columns are the same in all cases). This section gives examples of querying the WG15 recommended parameters and the following section gives examples of querying the other views.

1. **List the recommended analysis parameters for all the spectra of a given star**

   All the recommended parameters determined for stars in the archive are available in view 'RecommendedAstroAnalysis' (a view is an entity in a relational database which for the present purposes behaves like a table). The columns in view 'RecommendedAstroAnalysis' include 'cName' which lists the name of the star observed and 'wg' which lists the name of the working group that performed the analysis, in the form 'WGxx', for example 'WG10'. To list selected parameters (here effective temperature, log surface gravity, [Fe/H] ratio, Li I abundance and C II abundance) for all the spectra observed for a given star and the name of the working group that performed the analyses:

```
SELECT cName, wg, teff, logg, feh, li1, c1
FROM   RecommendedAstroAnalysis
WHERE  cName ='11053303-7700120';
```

   Note that two records are returned as this star has been analysed by WG11 and WG12; in general a given spectrum may be analysed by more than one working group.

2. **List the recommended analysis parameters for all the spectra available for a list of stars**

   Similarly to list selected parameters for all the spectra observed for a list of stars:

```
SELECT cName, wg, teff, logg, feh, li1, c1
FROM   RecommendedAstroAnalysis
WHERE  cName IN ('11034945-7700101', '11044460-7706240', '21101955-0200414')
ORDER BY cName;
```

   The list is ordered by 'cName' so that results for the same star appear together.

3. **List recommended analysis parameters alongside information about the spectrum from which they were obtained.**

   Aside on linking recommended parameters to their parent spectra

   To list analysis parameters determined from a spectrum alongside information about the spectrum it is necessary to join the 'RecommendedAstroAnalysis' and 'Spectrum' tables. However, these tables cannot be joined directly, but must be joined through the intermediate table 'SpectrumGroup'. The reason that 'SpectrumGroup' is needed is as follows. Usually recommended parameters are derived from a single spectrum. However, in some cases they are derived from two or more spectra, typically covering different wavelength regions. To cope with these latter cases the notion of the 'spectrum group' was introduced.

   A spectrum group is simply a unique group of spectra from which recommended parameters have been derived. Most spectrum groups will consist of a single spectrum, but some will consist of two or more. Each spectrum group has a unique identifier, which is tabulated in 'RecommendedAstroAnalysis' as column 'specGroupId'. Table 'SpectrumGroup' lists the spectrum identifiers of the spectra that constitute each spectrum group. If the spectrum group contains two spectra then the query will return two rows for each recommended parameter, one for each spectrum. It is a little easier to see what is going on if the 'specGroupId' and 'specId' identifiers are included in the listing (though, of course, they have no significance outside the GES archive):

```
SELECT
  spg.specGroupId, sp.specId,
  racc.cName, racc.wg, racc.teff, racc.logg, racc.feh,
  sp.expTime, sp.snr
FROM
  RecommendedAstroAnalysis racc,
  SpectrumGroup spg,
  Spectrum sp
```

```
WHERE racc.specGroupId = spg.specGroupId
  AND spg.specId = sp.specId
  AND racc.cName ='11053303-7700120';
```

In any event, the upshot is that the 'RecommendedAstroAnalysis' and 'Spectrum' tables must be linked using the 'SpectrumGroup' table.

To list analysis parameters determined from a spectrum alongside information about the spectrum it is necessary to join the 'RecommendedAstroAnalysis' and 'Spectrum' tables. However, these tables cannot be joined directly, but must be joined through the intermediate table 'SpectrumGroup', as explained above. So, for example:

```
SELECT
  racc.cName, racc.wg, racc.teff, racc.logg, racc.feh, racc.gratings,
  sp.expTime, sp.snr,
  spg.fileName
FROM
  RecommendedAstroAnalysis racc,
  SpectrumGroup spg,
  Spectrum sp
WHERE racc.specGroupId = spg.specGroupId
  AND spg.specId = sp.specId
  AND racc.cName ='11053303-7700120';
```

In this example in addition to the working group and some recommended parameters (effective temperature, log surface gravity and FeH ratio) the exposure time and signal-to-noise ratio of the spectrum from which they were derived are listed.

It is, of course, possible to display additional information about the spectra by joining the 'RecommendedAstroAnalysis', 'SpectrumGroup' and 'Spectrum' tables with the 'SpecFrame' table in the same way that the 'Spectrum' and 'SpecFrame' were joined (above). For example, to include the spectrograph used to acquire the spectra in the above query:

```
SELECT
  racc.cName, racc.wg, racc.teff, racc.logg, racc.feh,
  sp.expTime, sp.snr,
  frame.instrument
FROM
  RecommendedAstroAnalysis racc,
  SpectrumGroup spg,
  Spectrum sp,
  SpecFrame frame
WHERE racc.specGroupId = spg.specGroupId
  AND spg.specId = sp.specId
  AND sp.specFrameId = frame.specFrameId
  AND racc.cName ='11053303-7700120';
```

4. **List the recommended radial and rotational velocities for a given star.**

   See separate notes (which require login).

Queries on Working Group Recommended and Individual Node Analyses

As described in the previous section, each working group comprises several nodes and each node performs an independent analysis on some or all of the spectra (or rather spectrum groups) assigned to the group. The working group then combines these individual results to derive a set of astrophysical parameters and abundances recommended by the group. The individual nodes may perform a second set of analyses by adopting the astrophysical parameters recommended by the group and redetermining the abundances. Finally Working Group 15 uses the results of the various other groups to produce an overall recommended set of parameters (which are tabulated in view 'RecommendedAstroAnalysis' used in the previous section).

All these different types of result are stored in table 'AstroAnalysis' and there are views corresponding to each. The following table gives the details:

View	Description

RecommendedAstroAnalysis	Recommended, homogenised WG15 parameters and abundances.
WgRecommendedAstroAnalysis	Working group recommended parameters and abundances (for each working group)
WpNaAstroAnalysis	Individual node abundances determined using the parent working group's recommended astrophysical parameters.
NpNaAstroAnalysis	Astrophysical parameters and abundances determined by the individual nodes.
AstroAnalysis	All results.

Where a query of 'RecommendedAstroAnalysis' would yield a single row corresponding to the overall, WG15 recommended parameters for a spectrum (or rather a spectrum group) a query of 'WgRecommendedAstroAnalysis' would return several rows if the spectrum had been analysed by more than one working group. Similarly, a query of 'WpNaAstroAnalysis' or 'NpNaAstroAnalysis' would yield a set of rows, each row corresponding to an analysis by an individual node. The node responsible for the analysis is listed as column 'nodeName' (and working group recommended parameters have a 'nodeName' of the name of the working group). A query directly on table 'AstroAnalysis' would yield all the results for the spectrum group, though this is less likely to be useful.

As an example, to list selected parameters (here effective temperature, log surface gravity, [Fe/H] ratio, Li I abundance and C I abundance) recommended by the various working groups for all the spectra observed for a given star:

```
SELECT cName, wg, nodeName, teff, logg, feh, li1, c1
FROM   WgRecommendedAstroAnalysis
WHERE  cName ='11053303-7700120'
ORDER BY wg;
```

Three rows are returned listing values from WG12 and two analyses with a mixture of actual and default values from WG14. The selection can be further restricted to include just the results from a given working group:

```
SELECT cName, wg, nodeName, teff, logg, feh, li1, c1
FROM   WgRecommendedAstroAnalysis
WHERE  cName ='11053303-7700120'
  AND  wg = 'WG12';
```

The results from individual nodes can be shown by querying views 'WpNaAstroAnalysis' or 'NpNaAstroAnalysis'. For example, to show all the results for a given star where the nodes were computing both astrophysical parameters and abundances:

```
SELECT cName, wg, nodeName, teff, logg, feh, li1, c1
FROM   NpNaAstroAnalysis
WHERE  cName ='11053303-7700120'
ORDER BY wg, nodeName;
```

Note that the results are sorted by column 'wg' and then 'nodeName' so that the results for each working group appear together and within each working group the nodes are listed alphabetically. Exactly the same query can, of course, be made directly on table 'AstroAnalysis', though this is less likely to be useful:

```
SELECT cName, wg, nodeName, teff, logg, feh, li1, c1
FROM   AstroAnalysis
WHERE  cName ='11053303-7700120'
ORDER BY wg, nodeName;
```

In a similar fashion, all the examples in the previous section can be changed to query one of the other views by replacing view 'RecommendedAstroAnalysis' with the appropriate view name and adding the 'ORDER BY' clause.

Queries on Outlier Analyses

Three 'outlier' flags are present in table 'AstroAnalysis' and all the views on it ('RecommendedAstroAnalysis', 'WgRecommendedAstroAnalysis', 'WpNaAstroAnalysis' and

'NpNaAstroAnalysis'). Values of these flags indicate unusual features in one or more of the spectra on which the analysis was performed (and thus the spectrum deviates significantly from the norm for its spectral type and is an 'outlier'). The three columns of outlier flags are:

Column	Description	Code Range
peculi	Peculiarity flag(s)	1000-2999
remark	Spectral class flag(s)	3000-8999
tech	Technical issues flag(s)	9000-15000

For a given analysis each column can contain one or more alphnumeric flags. These flags are of the form:

numeric-code + alphabetic-confidence-designation

The range of numeric codes valid for each column is indicated in the table (above). The alphabetic confidence designation is a single letter as follows:

Confidence	
Designation	**Meaning**
A	probable
B	possible
C	tentative

If the value of a flag for a given row contains multiple entries then they are separated by a vertical bar ('|'). The WG14 Dictionary lists all the codes together with an explanation for each. A version with additional explanation is also available. Finally, the codes are also listed at the end of the WG14 Report.

For most analyses the three outlier columns are empty (most spectra are not outliers). However, the flags can be present in the analyses by individual nodes (views 'WpNaAstroAnalysis' and 'NpNaAstroAnalysis'), each working group's recommended values (view 'WgRecommendedAstroAnalysis') or the homogenised recommended values prepared by WG15 (view 'RecommendedAstroAnalysis'). In particular, WG14 only performed outlier analyses and did not otherwise analyse the spectra. WG14 comprised four nodes but these nodes did not submit individual analyses; rather they collated all their results into a single recommended set. The WG14 results are present in view 'WgRecommendedAstroAnalysis', just like the recommended values from other working groups, but the WG14 rows only have values for the columns identifying or characterising the spectra ('cName', 'targetID', 'fieldName' *etc.*) and the three flags, 'peculi', 'remark' and 'tech'.

The WG14 rows can be selected from view 'WgRecommendedAstroAnalysis' (or table 'AstroAnalysis') just like those for other working groups. *However, if you wish to use the WG14 flags in conjunction with the results of other working groups, including the WG15 homogenised values, then there is an important complication.* WG14 obtained its values by analysing *individual* spectra, *not* the groups of spectra (often a red, blue pair) used by the other working groups. Thus, to see which WG14 flags apply to which analyses by another working group it is not possible to join the rows using their 'specGroupID' (because the WG14 rows necessarily have a different 'specGroupID' from those of the other working group). Rather, the join must be made by going *via* the 'specGroupID' to obtain the 'specID' of each spectrum in each spectrum group and then joining these 'specIDs'. Two views have been defined to make using the WG14 results easier:

RecommendedOutlierAnalysis

> contains the WG14 flags joined to the WG15 flags for corresponding rows. A few other WG15 columns characterising the target and spectrograph setup (for example, 'cName', 'instrument', 'gratings' and 'fieldName') are also included.

SpectrumOutlierAnalysis

> exposes the 'specID' of each WG14 row and also all the populated columns, thus simplifying joins with other analyses.

Some simple examples of using these views follow.

1. **Which WG14 and WG15 flags are available for a given star?**

   The 'RecommendedOutlierAnalysis' view includes column 'cName' so simply select the row matching the required name:

   ```
   SELECT * FROM RecommendedOutlierAnalysis
   ```

```
WHERE cName = '08064390-4731532';
```

2. **How do the WG14 flags for a given star compare with those derived by another working group?**

   View 'SpectrumOutlierAnalysis' can be used to simplify joining the WG14 rows with those from another working group. For example to compare the WG14 and WG11 recommended flags for a given star:

```
SELECT
  wg14.*,
  wg11.peculi AS 'WG11_peculi',
  wg11.remark AS 'WG11_remark',
  wg11.tech AS 'WG11_tech'
FROM
  SpectrumOutlierAnalysis wg14,
  WgRecommendedAstroAnalysis wg11,
  SpectrumGroup sgp
WHERE wg11.wg = 'WG11'
  AND wg11.specGroupID = sgp.specGroupID
  AND wg14.specId = sgp.specId
  AND wg11.cName = '00194037-4659596';
```

   Note how here the red and blue spectra analysed individually by WG14 but as a pair by WG11 have resulted in two WG14 rows both joining with a single WG11 row to create two output rows. Similarly, it is possible to join the WG14 values to node, rather than recommended, rows:

```
SELECT
  wg14.*,
  np11.peculi AS 'Nodes_WG11_peculi',
  np11.remark AS 'Nodes_WG11_remark',
  np11.tech AS 'Nodes_WG11_tech',
  np11.nodeName AS 'WG11_Node_Name'
FROM
  SpectrumOutlierAnalysis wg14,
  NpNaAstroAnalysis np11,
  SpectrumGroup sgp
WHERE np11.wg = 'WG11'
  AND np11.specGroupID = sgp.specGroupID
  AND wg14.specId = sgp.specId
  AND np11.cName = '00194037-4659596'
ORDER BY np11.nodeName;
```

   Again each WG11 node analysis has paired with two WG14 rows. Finally, it is, of course, possible to perform joins directly using the WG14 rows in 'WgRecommendedAstroAnalysis' rather than 'SpectrumOutlierAnalysis', but there is no advantage in doing so and the join becomes more cumbersome:

```
SELECT
  wg11.cName,
  wg11.instrument AS 'WG11_instrument',
  wg11.gratings AS 'WG11_gratings',
  wg11.peculi AS 'WG11_peculi',
  wg11.remark AS 'WG11_remark',
  wg11.tech AS 'WG11_tech',
  wg14.instrument AS 'WG14_instrument',
  wg14.gratings AS 'WG14_gratings',
  wg14.peculi AS 'WG14_peculi',
  wg14.remark AS 'WG14_remark',
  wg14.tech AS 'WG14_tech'
FROM
  WgRecommendedAstroAnalysis wg11,
  WgRecommendedAstroAnalysis wg14,
  SpectrumGroup sgp11,
  SpectrumGroup sgp14
WHERE wg11.wg = 'WG11'
  AND wg14.wg = 'WG14'
  AND wg11.specGroupID = sgp11.specGroupID
  AND wg14.specGroupID = sgp14.specGroupID
  AND sgp11.specId = sgp14.specId
  AND wg11.cName = '00194037-4659596';
```

Queries on the Atomic and Molecular Line Lists

The GES archive contains a set of atomic and molecular line lists. Atomic lines both with and without hyperfine splitting are included. This dataset is described in the Linelist Report, which also discusses flags included in the tabulation that can be used to select suitable lines. These data are used consistently in all analyses of the GES spectra carried out within the GES Consortium. All the lines available are listed in table 'LineList', which contains in excess of 33 million rows, most of them corresponding to molecular lines. Several views containing useful sub-sets of the line list data are also provided. In many cases these views are likely to be more convenient to use than the complete 'LineList' table. The views available are as follows: (table 'LineList' is included for convenience and completeness):

View	Description	Rows
LineAtomHfs	Atomic lines including data for unresolved fine, hyper-fine or isotopic splitting, if present.	141,240
LineAtomNoHfs	Atomic lines excluding data for unresolved fine, hyper-fine or isotopic splitting.	139,950
LineMol	Molecular lines only.	35,003,307
LineMolAtomHfs	Molecular lines and atomic lines including data for unresolved fine, hyper-fine or isotopic splitting, if present.	35,144,547
(LineList)	The complete set of atomic lines with and without hyperfine splitting and molecular lines.	35,284,497

The following examples all present queries of view 'LineAtomHfs' (that is atomic lines with hyperfine splitting). However, all the views, and table 'LineList' have the same set of columns so the examples can be applied to any of them by simply substituting the appropriate view or table name.

1. **Which lines are available in a given wavelength range?**

   Column 'lambda' lists the wavelength in Ångström. Thus, to see all the lines in wavelength range 4800 to 4810 Å:

   ```
   SELECT * FROM LineAtomHfs
   WHERE lambda BETWEEN 4500 AND 4800
   ORDER BY lambda ASC;
   ```

   Note the inclusion of the 'ORDER BY' clause so that the lines are listed in order of increasing wavelength.

2. **Which lines are available for a given species?**

   Column 'name1' lists the element giving rise to each line and column 'ion' the ionisation state of each species. Thus, to see all the lines for singly ionised aluminium, ordered by increasing wavelength:

   ```
   SELECT * FROM LineAtomHfs
   WHERE name1 = 'Al' and ion = 1
   ORDER BY lambda ASC;
   ```

3. **Which species are available for a given element?**

   To see a list of the iron species available and the number of lines included for each species:

   ```
   SELECT name1, ion, count(ion) FROM LineAtomHfs
   WHERE name1 = 'Fe'
   GROUP BY name1, ion
   ORDER BY name1, ion;
   ```

4. **Which lines are available for a given element?**

   To see all the aluminium lines, for example, (including all the various ionisation states) ordered by increasing wavelength:

   ```
   SELECT * FROM LineAtomHfs
   WHERE name1 = 'Al'
   ORDER BY lambda ASC;
   ```