

## Exercise 1

Follow the “Whetting your appetite” demo from the lecture notes until you have the radius-luminosity plot for mcxc.

Now try the same thing for the eingalclus; when constructing the new query, you will have to change the columns to select and the table name.

Do the plots look similar? Use a plausible cross match (**Pair Match** in TOPCAT's **Joins** menu) to pair plausible clusters and plot radius and luminosity from the two catalogues against each other.

## Exercise 2

Continuing the last exercise, do the next steps for the mcxc example from the lecture notes until you can see the suspicious red objects. Take care that you do not lose track of which table you have selected at any time and make sure you adapt the table index in `TAP_UPLOAD.t4` to the index of your mcxc table. Can you find interesting objects? And can you think of ways to reduce the contamination with odd artefacts?

## Exercise 3

Follow the instructions in the SCS chapter (use the lecture notes) to get familiar with SCS and topcat.

1. Try to repeat the exercise using a Gaia catalogue instead of the HSOY. Caveat: the Gaia catalogue provides colours and magnitudes. Try to use the metadata of the table to figure out the names of the columns for brightness, and blue and red colours. Another good hint to not get lost in the column metadata is to search for “Gaia lite”.

In case you get stuck, you can also have a peek at:

<https://www.g-v-o.org/tutorials/pleiades.pdf>

2. Try to repeat the steps with the beehive cluster (“Praesepe”)

## Exercise 4

Use Aladin's viewer to study the object X Persei in different wavelengths. Get familiar how to select a survey and also make use of the data tree.

Hint: try the XMM survey. Try to guess what kind of object you are looking at. How could you confirm your hypothesis? (Well, of course there is Wikipedia, but what if you wanted to keep it in the VO family?)

## Exercise 5

Select the (rows of) the 20 brightest stars in the table `fk6.part1`.

## Exercise 6

Select the absolute magnitude and the common name for the 20 stars with the greatest visual magnitude in the table `fk6.part1` (in case you don't remember: The absolute magnitude is  $M = 5 + 5 \log \pi + m$  with the parallax in arcsec  $\pi$  and the apparent magnitude  $m$  (check the units!).

## Exercise 7

As before, select the absolute magnitude and the common name for the 20 stars with the greatest visual magnitude, but this time from the table `fk6.fk6join`. This will fail for reasons that should tell you something about the value of Bayesian statistics. Make the query work.

## Exercise 8

How many objects in the Fifth Catalogue of Nearby Stars  
(`cns5.main` on the GAVO TAP server) are missing a radial velocity?



## Exercise 9

Get the averages for the total proper motion from `lspm.main` in bins of one mag in `Jmag` each. Let the output table contain the number of objects in each bin, too.

## Exercise 10

Make an all-sky plot of the number of objects and their average effective temperature in HEALPixes of level 5 of the catalogue `rave.main`. Hint: In the server-provided [Examples](#) on the GAVO server, there is an example “Make a HEALPix Map of something” (in [Local UDFs](#); if you don’t see it, update your TOPCAT). Start from there.

Can you understand the structures that you see?

## Exercise 11

Look at the documentation of the `ivo_epoch_prop_pos` UDF (refer back to the UDF slide if necessary). Can you figure out how to propagate (i.e., apply the proper motions to compute positions in the future) the CNS5 to the year 2150? The positions in the CNS5 are (somewhat unusually) given for what is in the column `epoch`.

What's the RA of Sirius you determine in this way? And why will this be probably a rather poor guess?

## Exercise 12

Compare the radial velocities given by the `rave.main` and `arihip.main` catalogues, together with the respective identifiers (`hipno` for `arihip`, `raveid` for `rave`). Use the `POINT` and `CIRCLE` functions to perform this positional crossmatch with, say, a couple of arcsecs.

## Exercise 13

Sit back for a minute and think whether the JOIN and the EXIST solution in the *Subqueries* chapter are actually equivalent. You are not supposed to see this from staring at the queries – but comparing the results from the two queries ought to give you a hint; retrieve a few more objects if your results happen to be identical.

## Exercise 14

If you have some data with celestial positions of your own, try reading it into TOPCAT and try the crossmatch with that. If you do not have any suitable data, try the ex.vot from the *TAP*: *Uploads* slide.

## Exercise 15

Follow the example on the “Almost Real World” slide with the matchme.vot table provided there.

Despite the artificial setting, we have lost one object in the upload join. Can you find it? And can you guess why we have lost it?

Hint: Have a look at TOPCAT's **Pair Match** facility, paying attention to the **Join Type** setting.

## Exercise 16

In the last exercise, we met the star with the Gaia source id 1872046574983497216 and a total proper motion of 5 arcsec/yr. In the solution I claimed this is a really extreme case. Well: how extreme is it? Can you estimate how many faster stars there are? (Please resist the temptation to use the full Gaia catalogue for this purpose; see also the next exercise).



## Exercise 17

(This is slightly advanced) In the last exercise, you were asked not to consult the Gaia source catalogue to get proper motion statistics, although to a contemporary astronomer that would be the obvious choice. That is because all-catalogue statistics are expensive on Gaia.

Can you find a way to still get the fastest stars in `gaia.dr3lite` within the time limit of sync queries on that server (i.e., a couple of seconds)?

Cheap hint: see what columns are indexed.

## Exercise 18

In async mode, run this on the GAVO server:

```
SELECT TOP 500 source_id, flux  
FROM gdr3spec.spectra  
WHERE arr_max(flux)>arr_avg(flux)*5
```

This is using the experimental array extension to ADQL<sup>1</sup>. You can probably guess without reading the blog post that this will select spectra with something like strong lines.

Run that query in async mode on the GAVO server. In a course situation, shout out your job's phases to watch the dequeuing. Save the job URL, exit TOPCAT, resume it, and load the result when the job is COMPLETE-d.

<sup>1</sup><https://blog.g-vo.org/a-proposed-vector-extension-for-adql.html>

## Exercise 19

Pick a server that piques your interest from TOPCAT's server selection. How many tables are there on the server? How many columns? How many columns with UCDs starting with phot.mag?

## Exercise 20

In exercise 18, you selected stars with odd spectra. Can you use Simbad's TAP service to find what types of star these are?

Hint: you probably need to do two upload joins, first with `gaia.dr3lite` (or some other Gaia DR3 table out there), then with `public.basic` on Simbad.

## Exercise 21

Plot the total coverage of the Lockman Hole Radio Survey in the table `emi.main` on the GAVO TAP server as a level 8-MOC in TOPCAT. What is its area? Hint: you will probably need a subquery.

For aesthetic reasons, also try this at levels 12 and 18.

## Exercise 22

Get our example `basicsiap.py` from the notes.

Now find an image service publishing the ROSAT survey and pointed observations and see if it has an image for the position given (or try some other service and position you are actually interested in).

Use [WIRR](#) to search the VO Registry for now.

What is coming back from `SIAService`'s search is a sequence of `SIASRecords`. Have a quick look at [its pyvo documentation](#) and make your program print the file size and the instrument name rather than calling `cachedataset`.

## Exercise 23

Get the `globalsiap.py` script from the attachment and change it so it skips 90% of the services discovered randomly (use `random.random()`). Also, remove the constraint on the date (we don't need that here) and change the position to something you are interested in or expect to have pretty pictures (M1 or M51 are always good candidates). Run the thing and see what you find.

## Exercise 24

Get the pyVO source code and find the source of `pyvo.samp`. Start TOPCAT, find the implementation of the `connection` context manager, and then open a SAMP connection manually from an interactive Python prompt. And then again, and a third time. What do you observe in TOPCAT?

Hint: To get the source code, try:

```
git clone https://github.com/astropy/pyvo.
```

Or, on Debian-derived boxes:

```
apt source python3-pyvo
```



## Exercise 25

Still in `samp.py`, inspect how `send_image_to` is implemented. From reading the code, can you figure out how to only send the image to Aladin? If you can, try your solution in `globalsiapsamp.py` by having Aladin and ds9 (Debian package: `saods9`) open at the same time.

Hint: To find out Aladin's client name, check TOPCAT's SAMP status window.

## Exercise 26

Write a program that prints the number of rows in the table `arihip.main` in the TAP service at <http://dc.g-vo.org/tap> (do *not* pull all the rows and use python's `len`).

Hint: With ADQL's `AS` construct you can control the names of table columns.

## Exercise 27

The following program should print URLs and titles for images in some collection for whatever names are in `OBJECTS`:

```
import pyvo

OBJECTS = ["IC 4756", "NGC 3377"]
QUERY = """select accref, imagetitle
           from maidanak.reduced
           where object={object}"""

svc = pyvo.dal.TAPService("https://dc.g-vo.org/tap")
for object in OBJECTS:
    print(svc.run_sync(QUERY.format(**locals()))).to_table()
```

What really happens: An error message. Can you figure out where it comes from and how to fix things?

## Exercise 28

Use TOPCAT's TAP data browser to locate services and table names for TGAS and RAVE. Also figure out where the positions and some usable magnitude are, plus the proper motions from TGAS and the radial velocities from RAVE.

Re-write `fetch3.py` to query the retrieve all stars between 8 and 8.2 mags from each table. Also, send the results to Aladin (which is known as *Aladin* (capitalised) on the SAMP bus). See if you can get a nice plot of `rv`, `pmra`, and `pmdec`.

Hint: Check Aladin's [Catalog/Create filter](#) for fancy plotting options.

## Exercise 29

Go through the source code of `fetch3-cluster.py`. You will see we have put in two workarounds for where the data providers messed up. Can you see in each case what might have gone wrong? Have the service operators fixed their software or do things still fail when you remove a workaround? In a course setting, coordinate with your neighbours and split up the work so each only looks at one workaround.

## Exercise 30

Run `fetch3-cluster.py` and select a couple of objects. Keep the resulting file (`selected_positions.vot`) – we will want to reuse it later.

## Exercise 31

You can use URLs in a query's upload argument. To try this out, review the TGAS and RAVE exercise [28](#). Let the initial RAVE query be asynchronous. On the resulting job, call `wait` as above. Once it is done, upload what is job's `result_uri` attribute into the TGAS server with a normal positional upload join.

## Exercise 32

Can you change `get_spectra.py` such that only spectra of resolving power 10000 or greater are retrieved?

Hint: Use TOPCAT or the `tables` property of your `TAPService` to inspect the metadata of the `ivoa.obscore` table to figure out which column to query against. Just in case: It is almost always better to filter on the remote side rather than the local side. And chuck the “almost” if the constraint can be expressed as a single condition in a `WHERE` clause.



## Exercise 33

Can you figure out the default output limit (i.e., in effect an implied TOP) for the TAP service at <http://dc.g-vo.org/tap>? How far can you raise it?

Can you write a program that figures it out for all TAP services out there that talk about tgas?

## Exercise 34

Which IAU constellation is the least massive exoplanet in the exoplanet merged catalogue in? Try solving this using pyVO's registry API; hint: to figure out constellations, having the constellations as ADQL polygons is really handy.

## Exercise 35

(You will need to have looked at the vocabularies sidetrack for this)

Take `new-constraint.py` and add support for query expansion: add a keyword argument `expand`. If that is true, include the narrower concepts of what was passed in, too.

Hint: You can leave (something like) this to the server with a UDF, or you can do the query expansion locally; the first way is simpler, the second perhaps more instructive.

## Exercise 36

Write a function `get_available_semantics(dl) -> set` returning a set of the semantics available for a given datalink.

Try your program on the SSA example from the lecture.

## Exercise 37

Get the `soda-with-rows.py` script for doing cutouts on CALIFA DR3 and make a false colour image for IC 1151 by taking the slices from the COMB cube (see the setup column) at 400 nm as blue, at 550 nm as green, and at 700 nm as red. Do not download the whole cube, use SODA to just retrieve exactly what you need.

## Exercise 38

The action of the SAMP handler is in the `make_response_table` method; have a brief look at it to appreciate what is going on. Then, replace what is there with something that does a SIAP search on the service at <http://dc.g-vo.org/lswscans/res/positions/siap/siap.xml> and returns the corresponding table for sending to Aladin (hint: remember the `to_table` method of DAL results).

## Exercise 39

Listening to the SAMP message *coord.pointAt.sky*, implement an “odometer” computing and printing after each step the distance travelled by the pointer.

To do this, you will need to keep the SAMP connection, the last position and the distance travelled so far as state; take the *vicinitysearcher*, remove the code keeping the state and behaviour used for its function, and insert our new logic.

Hints: Look at *SkyCoord* in *Astropy* and the *mtypes* page; when re-using SAMP bindings, make sure you handle messages, not calls.

## Exercise 40

In `multitap.py`, have a look at `get_services_and_tables`; in there, we are doing a grouping operation on the client (i.e., our) side. Can you move to to the server side using `GROUP BY` and the `ivo_string_agg` UDF?



## Exercise 41

Can you find out the strings you need to pass to `get_feature` find  
find out whether a service supports the nifty `IN_UNIT` function?

## Exercise 42

There is one glaring hole in our multitap script: Units. Try to improve on this: If the service supports `IN_UNIT`, use it in about the way we have been using `CAST`.

If you actually need something like this, you can of course also compute the conversion factors locally (using `astropy.units`) and bake them into the queries. Feel free to try that, too.

## Exercise 43

Get the `epnquery.py` and change it to only discover spectra (that's dataproduct type *sp* in EPN-TAP). then send the first two spectra your program finds to TOPCAT (or SPLAT, or CASSIS, if you have one of them).

## Exercise 44

The SSAP service at <http://dc.g-vo.org/theossa/q/ssa/ssap.xml?> houses theoretical spectra mostly of hot, compact stars.

See if you can retrieve three spectra for stars with `log_g` between 4.5 and 5.5, an effective temperature between  $7 \times 10^4$  and  $10^5$  Kelvin, and a Nitrogen mass fraction larger than 0.015 dex (write `+Inf` for “no upper limit”).

Send the spectra retrieved to `splat`.

Hints: Use `viewparams.py`, start from `siapectra.py`, remember `dal.ssa.SSAService`, and pass in `FORMAT='VOTable'` to avoid retrieving spectra in both FITS and VOTable.

## Exercise 45

Add full Gaia records from `ivo://esavo/gaia/tap's DR3` `gaia_source` to some records from the `hdgaia.main` table on GAVO's data centre. This does not need any slicing; still, only upload what you actually need for matching; for that, the `smart-tap-upload.py` example should be helpful.

Hint: for our simple `table.join` to work (which needs the same name in both tables), it is probably smart to rename `source_id3` in `hdgaia` at the ADQL level.

## Exercise 46

Assume you are about to publish a table containing a column that gives the angular size of an object you observed. What would be a good UCD to assign to that column?

## Exercise 47

Assume you are about to publish a table containing a column where you subtracted magnitudes (or the same object, of course) in the SDSS u and r bands. Can you come up with a good UCD for that?

## Exercise 48

The constraint

```
1=gavo_vocmatch('product-type',  
  'spatially-resolved-dataset', dataproduct_type)
```

that we have put in on the vocabularies in ADQL slide claims to match *spatially-resolved-dataset* and all narrower concepts. Can you give an equivalent expression of the form

```
dataproduct_type IN ('...', ....)
```

based on the product-type vocabulary? And why is that less desirable than using the UDF?



## Exercise 49

Get the VOTable at <http://dc.g-vo.org/arihip/q/cone/scs.xml?RA=333&DEC=43&SR=2&RESPONSEFORMAT=votable> and add to the value of the publisher *INFO* a (alas, hypothetical) **(note to self: they were on holiday in May 2024)**, using a text editor. If you have xmlstarlet, try re-formatting it first.

Ensure that the edit actually happened using TOPCAT. There, edit the note, too (doubleclick) and then see if you can see the change in the text editor.

## Exercise 50

Again in our VOTable, use a text editor to add an *INFO* element with a name of **(yourname)-note**, a value of **(today's date): learned how to add INFO elements the hard way**, and a content of **private processing note**. Load your modified table into TOPCAT to ensure you have not damaged the file and the information is there.

## Exercise 51

Still in our VOTable, set the `vrad` field for the object with the Hipparcos number 109481 to the value  $-16.268137$  (which is what Gaia DR3 gives for this object).

Again, try it once with TOPCAT and once with a text editor. For the latter, you will need the table to be in *TABLEDATA* format. At your option, use the `RESPONSEFORMAT` parameter or just save the table as *TABLEDATA* from TOPCAT.