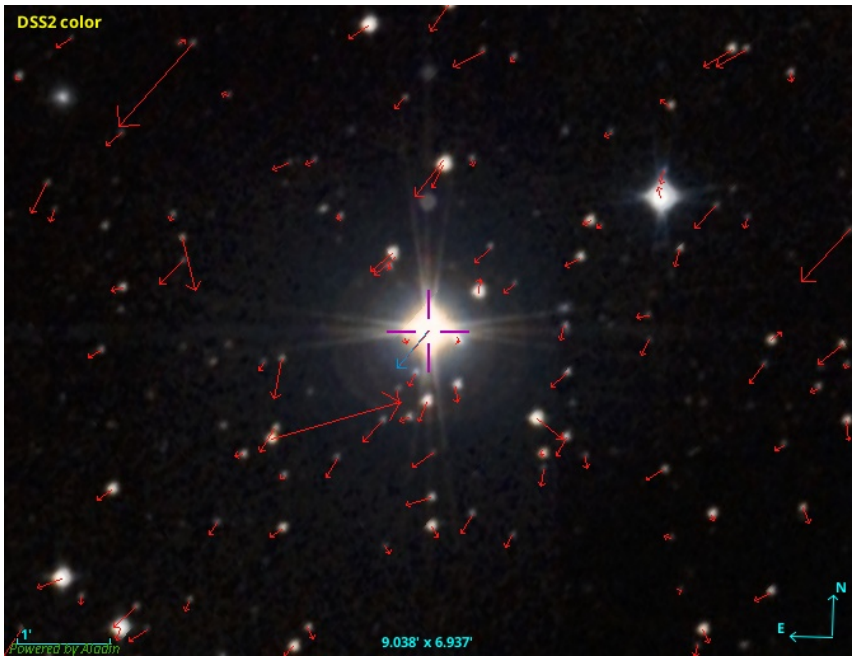
Hint Sheet For The 2026 GAVO Puzzler



Markus Demleitner

September 10, 2026

Abstract

The 2026 GAVO puzzler, a little challenge to flex your Virtual Observatory skills (or make you want to develop them), asks: How bright is the brightest star that has changed its (IAU) constellation since 2000? These hints let you figure it out by developing a somewhat non-trivial ADQL query.

Software: TOPCAT

1 Starting out

While there's nothing wrong with using another TAP client (and if you know your way around with pyVO, these hints would make a nice notebook), here we assume you have TOPCAT. You should not be without it anyway, notebooks notwithstanding. So please fetch it and install it on your machine: `https://www.star.bris.ac.uk/~mbt/topcat/` (or `apt install topcat` on open systems).

2 Finding constellation data

While for positions and proper motions these days it's safe to just go to Gaia's source table, constellation data is a bit more elusive. To find some, in TOPCAT hit VO → Table Access Protocol – whenever there's a non-trivial discovery problem, TAP is probably a very suitable tool.

Type `constellations` into the Keywords field and hit return. You will find some matches in VizieR and one in the GAVO DC TAP service. That latter talks about "ADQL-queriable polygons", which translated from VO jargon means: "I want that".

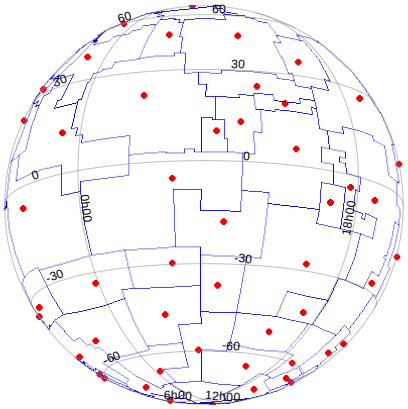


Figure 1: TOPCAT's rendering of GAVO's constellation polygons.

So, double click the table. You will end up in the Use Service tab and can inspect the table metadata. Have a brief look. You can make a quick plot of the constellations by saying:

```
SELECT * FROM cstl.geo
```

(you can make TOPCAT paste in the table name by hitting the TBL button right above the query field). Then, in TOPCAT's main window, click Graphics → Sky Plot, in the new window Layers → Add Area Control and then in the lower pane, select the table you just downloaded (something with "cstl.geo" in it, presumably). TOPCAT figures out the rest, and you will see something like Fig. 1.

3 Finding jumping stars

▷ **1** *Find a suitable Gaia Table* – It's always nice if you do not have to span servers with your queries, so let's see if your service has Gaia data. To do that, type `gaia` into the Find text box. You will find gaia.dr3lite, and if you inspect its metadata you will see: Yes, that will do, we have positions and proper motions here.

▷ **2** *Make a play set* – Working with the full 1.8 billion sources is going to be slow. Once everything is done, runtime is less of an issue, but when you develop ADQL queries, always pick a small subset. Here, perhaps the naked-eye stars would do? See how many there are in the Gaia source list:

```
select count(*)
from gaia.dr3lite
where phot_g_mean_mag<6
```

Double-click the result table in TOPCAT's main window; you will see that's 6764 objects, which is a nice size to play with.

▷ **3** *Select the columns you want to work with* – For what we want to do, we will need positions, proper motions (we'll be too lazy to care about errors), parallax and RV for exact epoch propagation, and the magitude. To select the columns, type `SELECT` into the query box, go to the Columns tab, and holding the control key click the various columns that you want. and then click COLS above the query box. The result would be something like

```
SELECT ra, dec, pmra, pmdec, parallax, phot_g_mean_mag, radial_velocity
```

(you can also type this in, but this is a case where clicking usually is more fun).

Complete the query with what you had above:

```
from gaia.dr3lite
where phot_g_mean_mag<6
```

You could run the resulting query, but it's not really worth it.

▷ **4** *Add propagated positions* – Fasten your seat belts; what's following probably requires a bit of ADQL experience to understand. To propagate the Gaia positions (which are given for 2016.0) to 2000 and 2026, you could just add the proper motions (be careful with the $\cos(\delta)$ in RA, though) times the epoch difference to the coordinates; this would be good enough given the small movements we talk about here. However, it's much safer to use exact propagation. In the ADQL tab, under Service-specific UDFs, you will find `ivo_epoch_prop_pos` that tells you how to use this. Or just trust us.

What we want is to add fields to our sample. In classic ADQL, this would be a subquery. In modern ADQL, there are CTEs, Common Table Expressions. These let you write queries in steps that are, to my taste, much easier to follow. Consider this:

```
with cands as (
  select source_id, phot_g_mean_mag, ra, dec,
    parallax, pmra, pmdec, radial_velocity
  from gaia.dr3lite
  where phot_g_mean_mag<6),

proppos as (select source_id, phot_g_mean_mag,
  ivo_epoch_prop_pos(ra, dec, parallax, pmra, pmdec,
    radial_velocity, 2016, 2000) as pos2000,
  ivo_epoch_prop_pos(ra, dec, parallax, pmra, pmdec,
    radial_velocity, 2016, 2026) as pos2026
  from cands)

select * from proppos
```

Can you see how we go from one table to the next, step by step, which makes our eventual "main" query trivial in this case? Feel free to run this query to get a better idea of what is going on.

▷ **5** *Add constellation info* – Remember our constellation table? Turning the positions we just computed into constellations now means joining with cstl.geo; this is a bit tricky because we have to join the table twice, once joining on pos2000, and once on pos2026. But it's just another CTE. Try adding a comma after `cands)` above and then adding:

```
withcstl as (
select  source_id, phot_g_mean_mag,
  c0.name as cstl2000, c6.name cstl2026
from proppos
join cstl.geo as c0 on 1=contains(pos2000, c0.p)
join cstl.geo as c6 on 1=contains(pos2026, c6.p))
```

Can you make out how we are using aliases to have the constellation names for both 2000 and 2026 in one table?

If you run the resulting query, your result will be unchanged, because you will still be selecting from the proppos CTE. Change proppos to withcstl and you will see constellation names – unsurprisingly mostly the same.

▷ **6** *Select the jumping stars* – That's been the heavy lifting. To filter for jumping stars, just see where cstl2000 is different from cstl2026. Your full query then is

```
with cands as (
  select source_id, phot_g_mean_mag, ra, dec,
    parallax, pmra, pmdec, radial_velocity
  from gaia.dr3lite
  where phot_g_mean_mag<6),

proppos as (select source_id, phot_g_mean_mag,
  ivo_epoch_prop_pos(ra, dec, parallax, pmra, pmdec,
    radial_velocity, 2016, 2000) as pos2000,
  ivo_epoch_prop_pos(ra, dec, parallax, pmra, pmdec,
    radial_velocity, 2016, 2026) as pos2026
  from cands),
```

```
withcstl as (
select  source_id, phot_g_mean_mag,
  c0.name as cstl2000, c6.name as cstl2026
from proppos
join cstl.geo as c0 on 1=contains(pos2000, c0.p)
join cstl.geo as c6 on 1=contains(pos2026, c6.p))

select * from withcstl
where cstl2000!=cstl2026
```

The result is, unsurprisingly, empty. No bright star has jumped constellations.

▷ **7** *Try with a fainter magnitude limit* – So, let's include more stars: we won't transfer large amounts of data even if we go down to, say, 12th magnitude. To keep runtimes in the just-for-fun range, try a limit of 10. Even that will give you a timeout when you query synchronously (you're epoch-propagating 0.5 million objects spread all over the disk twice). Hence, switch Query Mode to `Asynchronous` right above the query box to run that full query. The result should be in within about a minute.

Find me on VO Text Treasures http://dc.g-vo.org/VOTT

This document is in the public domain.