SQL Queries

Modern astronomical surveys

- With the advent of the digital age, driven by the use of CCDs in cameras, astronomical surveys have started to become semi or fully automated
- So, huge amounts of data are now arriving from sky surveys (Tb=Terabyte, Pb = Petabyte = 1000Tb)
  - ~50Tb of reduced data products over ~10 years of the Sloan Digital Sky Survey (SDSS)
  - ~2Pb of reduced data products over ~5 years of the Dark Energy Survey (DES)
  - ~50Pb of reduced data products over ~5 years of LSST (Large Synoptic Survey Telescope) operations

Mining modern astronomical surveys

- With such a large amount of data to sift through, astronomers have become more involved in developing data mining techniques
- We've discussed aspects of this in terms of pixelating the sky...which is really a method for indexing large amounts of data in a database for efficient searches
- The HTM index, a type of quad-tree that we've discussed briefly, is an efficient schema for storing data and searching through that data by object position
- We won't discuss the math of HTM in detail (a good description is linked from the syllabus) but think of it as a HEALPix-like index, coupled with the spherical cap formalism to find which HTM pixels lie in a cap

Introduction to SQL

- Visit the *SDSS SQL Tutorial* linked from the syllabus
- Read and/or try the following tutorials:
  - *1. Introduction*
  - *2. A simple Query*
  - *3. Common Searches*
- Note though, that nothing in these first 3 SQL tutorials makes use of the HTM indexing scheme
- The genius of HTM is coded in functions such as, e.g., *fGetNearbyObjEq*(α,δ,θ) which can *very* rapidly find objects at a radius θ around a position (α=RA, δ=dec)
  - Try *SDSS SQL Tutorial 10. Functions*

Python tasks

1. Using the *SDSS SQL Search Box* (linked from the syllabus) download RA, Dec and g-band imaging for all objects in the SDSS within θ =2' of (α,δ) = (300°,-1°)
2. Write Python code that reads in these SDSS objects and plots RA against dec. *Use circles for your data points*
3. Repeat your plot, but bin your points such that objects with *larger* g are plotted using *smaller* circles (i.e. plot $16 < g < 17$ at one size, plot $17 < g < 18$ at smaller size)
   - Using Matplotlib, *plt.scatter(ra,dec,s=s)* will allow you to plot points of different sizes
   - Here, *s* is the "size" of the point, but note that it's actually the *area* of the marker (so multiplying *s* by 4 will double the radius of the plotted point)

Python tasks

4. Use the *SDSS Navigator Tool* linked from the syllabus, to display the SDSS image near (α,δ) = (300^o,-1^o). Zoom in until the image is a few arcminutes across (the scale will be about 20")
   - Check that your plot looks reasonably like the *SDSS Navigator Tool* image