SQL Queries for Target Selection For *Breakthrough Listen*'s MeerKAT Commensal Observing Campaign

Logan Pearce

August 11, 2018

1 Introduction

This document is a compilation of the SQL and ADQL commands and methods I used to generate the BL MeerKAT 1-million star target list.

2 Example Queries

<table>
<tr><th colspan="2">MySQL and BL databases</th></tr>
<tr><th colspan="2">Connecting to the database through Python</th></tr>
<tr><td>Using the Python package MySQLdb
For now the target lists are hosted in the "loganp" database
"Read default file" sources the username, password, host address, and database to query from the .my/.cnf file. Username and password can be manually entered by using "user= " and "password= "
"autocommit=True" allows all future cursor commands to be committed automatically without requiring a separate commit step
"local infile = 1" allows commands referring to local files to be loaded correctly.</td><td><pre>import MySQLdb
db = MySQLdb.connect
(
 read_default_file="~/.my.cnf",
 autocommit=True,
 local_infile = 1
)</pre></td></tr>
<tr><td>Using the Python package MySQL Connector/Python
MySQL Connector/Python is a pure-python implementation created by Oracle, so it is typically considered the preferred method of connecting. Everything downstream of the connection behaves the same way whether connecting through MySQLdb or Connector/Python. This examples performs the same function as the MySQLdb example using the Connector/Python syntax.</td><td><pre>import mysql.connector
db = mysql.connector.connect
(
  option_files='/Users/loganpearce/.my.cnf'
  allow_local_infile=True,
  autocommit=True
)</pre></td></tr>
<tr><th colspan="2">Sending generic SQL commands
You can send any SQL command to the server through use of a cursor.</th></tr>
<tr><td>Define cursor</td><td><pre>cursor=db.cursor()
cursor.execute(COMMAND)</pre></td></tr>
</table>

Create new table	`command =`
	`"CREATE TABLE new_table (`
	`column1 INT(8),`
	`column2 VARCHAR(255),`
	`column3 FLOAT,`
	`....`
	`);"`
	`cursor.execute(command)`

Create new table with the same properties as another table
This will copy over all the columns from the old table without copying the data. Useful if you have a lot of columns or several similar tables to make. These tables have 57 columns.
	`command =`
	`"CREATE TABLE ‘1M_target_list‘`
	`LIKE master_gaia_database;"`
	`cursor.execute(command)`

Copy all or some of the data from one table into another table
This command copies only sources from the master table that fall within a specified RA and Dec range.
	`string = "INSERT INTO meerkat_gaiadr2_laduma`
	`SELECT * FROM master_gaia_database`
	`WHERE ‘ra‘ BETWEEN 52.75 AND 53.375`
	`AND ‘decl‘ BETWEEN -28.0 AND -27.6;`
	`cursor.execute(string)`

Add a new column to existing table	`string = "ALTER TABLE ‘1M_target_list‘`
	`ADD ‘dist_c‘ FLOAT NOT NULL;"`
	`cursor.execute(string)`

Update a table entry with new data
This command sets the distance entry in the LADUMA table to a value of 200 for the item in index 12
	`string = "UPDATE ‘1M_target_list‘`
	`SET ‘dist_c‘=200`
	`WHERE ‘index‘ = 12;"`
	`cursor.execute(string)`

Drop a column from table	`string = "ALTER TABLE ‘1M_target_list‘`
	`DROP COLUMN ‘dist_c‘;"`
	`cursor.execute(string)`

Remove all data from table but leave the table structure intact	`string = "DELETE FROM ‘1M_target_list‘`
	`WHERE 1;"`
	`cursor.execute(string)`

Querying database to return results	

Use the cursor
This method works, but it can be a bit tricky to get the results from the cursor object.
	`string="SELECT ra,decl`
	`FROM ‘1M_target_list‘`
	`WHERE ‘dist_c‘ > 100 "`
	`cursor.execute(string)`

Use Pandas
Use the Pandas package to query the database and return the results in an easy to manipulate Pandas dataframe.
	`import pandas as pd`
	`string="SELECT *`
	`FROM ‘1M_target_list‘`
	`WHERE 1 "`
	`df = pd.read_sql(string, con=db)`

Example queries	

Select all objects in an RA/Dec range	`SELECT *`
	`FROM master_gaia_database`
	`WHERE ra BETWEEN 0.0 AND 90.0`
	`AND decl BETWEEN -50.0 AND 20.0;`

Select all objects out to a certain distance range	`SELECT *`
	`FROM master_gaia_database`
	`WHERE ‘dist_c‘ < 100;`

Select only some columns	`SELECT ‘ra‘,‘decl‘,‘parallax‘,‘dist_c‘`
	`FROM ‘1M_target_list‘`
	`WHERE ‘dist.c‘ < 100;`

Select all objects within a circle of radius 0.5 from a specified RA/Dec	`SELECT * FROM ‘1M_target_list‘`
	`WHERE POWER((ra-(15.0)),2) +`
	`POWER((decl - (-20.0)),2) < 0.25;`

Select all SpType G objects	`SELECT * FROM ‘1M_target_list‘`
	`WHERE ‘sptype_c‘`
	`LIKE ’%G%’`

ADQL and Gaia databases	
Connecting to Gaia database through Astroquery TAP+	`from astroquery.gaia import Gaia`

Launch asynchronous query
 	
This query finds the top 10 highest parallax objects in the Gaia DR2 source catalog with parallax less than 800, and dumps the results to a csv file.
	`query = "SELECT TOP 10 * FROM gaiadr2.gaia_source`
	`  WHERE parallax <= 800`
	`  ORDER BY parallax DESC"`
	`job = Gaia.launch_job_async`
	`  (`
	`  query=query,verbose=False,`
	`  dump_to_file=True,`
	`  output_format='csv'`
	`  )`

Perform query with "on the fly" uploaded table	`upload_resource = 'my_table.xml'`
	`j = Gaia.launch_job(query="SELECT *`
	`  FROM tap_upload.table_test",`
	`  upload_resource=upload_resource,`
	`  upload_table_name="table_test",`
	`  verbose=True)`
	`r = j.get_results()`

Log in to query	`Gaia.login(user='userName', password='userPassword')`
	`Gaia.login(credentials_file='my_credentials_file')`
	`Gaia.logout()`

Get list of available public tables	`tables = Gaia.load_tables(only_names=True)`
Example queries unique to ADQL	

Cone search around a specific point with a 5 arcsecond search radius. The syntax within the CIRCLE condition is CIRCLE('coord sys',RA,Dec,Search Radius), all in degrees.	`SELECT *`
	`FROM gaiadr2.gaia_source`
	`WHERE CONTAINS(POINT(`
	`  'ICRS',gaiadr2.gaia_source.ra,`
	`  gaiadr2.gaia_source.dec),`
	`  CIRCLE('ICRS',0.004167,-19.498611,0.0014))=1;`

Object query	`coord = SkyCoord(ra=280, dec=-60,`
	`  unit=(u.degree, u.degree), frame='icrs')`
	`width = u.Quantity(0.1, u.deg)`
	`height = u.Quantity(0.1, u.deg)`
	`r = Gaia.query_object_async(`
	`  coordinate=coord, width=width, height=height)`

AQDL query for high-quality sources in Gaia catalog

```
SELECT * FROM gaiadr2.gaia_source
WHERE parallax_over_error > 20
AND phot_g_mean_flux_over_error>50
AND phot_rp_mean_flux_over_error>20
AND phot_bp_mean_flux_over_error>20
AND phot_bp_rp_excess_factor < 1.3+
  0.06*power(phot_bp_mean_mag-phot_rp_mean_mag,2)
AND phot_bp_rp_excess_factor > 1.0+
  0.015*power(phot_bp_mean_mag-phot_rp_mean_mag,2)
AND visibility_periods_used>=8
AND astrometric_chi2_al/(astrometric_n_good_obs_al-5)
  <1.44*greatest(1,exp(-0.4*(phot_g_mean_mag-19.5)))
```

Other Astroquery Queries

Querying the NASA Exoplanet Archive

```
from astroquery.nasa_exoplanet_archive
      import NasaExoplanetArchive
t = NasaExoplanetArchive.get_confirmed_planets_table()
h = NasaExoplanetArchive.query_planet('HAT-P-11 b')
```

Querying the NASA Exoplanet Orbit Database

```
from astroquery.exoplanet_orbit_database
     import ExoplanetOrbitDatabase
t = ExoplanetOrbitDatabase.get_table()
h = ExoplanetOrbitDatabase.query_planet('HAT-P-11 b')
```

Querying SIMBAD

Query 1 object
Query Messier objects 1-9
Cone search around an object
Cone search around a coordinate
Search just a specific catalog
Query bibcode
Retrieve all the names associated with an object
Retrieve all the objects contained in an article

```
from astroquery.simbad import Simbad
t = Simbad.query_object("GSC 6214-210")
r = Simbad.query_object("m [1-9]", wildcard=True)
import astropy.units as u
r = Simbad.query_region("m81", radius=0.1 * u.deg)
import astropy.coordinates as coord
r = Simbad.query_region(coord.SkyCoord("05h35m17.3s
      -05h23m28s", frame='icrs'), radius='1d0m0s')
r = limitedSimbad.query_catalog('eso')
r = Simbad.query_bibcode('2005A&A.430.165F')
r = Simbad.query_bibcode('2013A&A*', wildcard=True)
r = Simbad.query_objectids("Polaris")
r = Simbad.query_bibobj('2006AJ....131.1163S')
```

Astroquery "Gallery of Queries"
https://astroquery.readthedocs.io/en/latest/gallery.html

Astroquery list of available databases for queries
https://astroquery.readthedocs.io/en/latest/index.html