AutoQuery:

Autoquerying is the process where we create a recognition/association matrix that represents the relationships between chips. We begin by generating a matrix by querying every chip against other chips. When each chip is queried we have to compare that one chip to all other chips in the chip table. This was set up so that it would be in parallel, however this was giving us many errors so we decided to move to doing this in series. We simply forced HotSpotter to perform computations in series, thus serving as a sort of band-aid. We have changed the file structure somewhat since this change, so there is a chance that parallel feature computation is possible now.

To automate HotSpotter we created created functionality in autoquery.py that walks through the whole chip table and queries a chip against every other chip. This is done using functions listed in the MCL/Parsing ICD. However if there are no keypoints in a chip then it will return a string for the match between the two chips. We correct for this by assigning all strings to zero. HotSpotter however does not give the same scores from chip to chip. For example when querying chip 1 we might find chip 2 matches with a score of 500, but when chip 2 is queried we may find that chip 1 gets a matching score of 50. To cluster these chips we need an undirected graph. We create an undirected graph by averaging the scores from the queries and putting them into the score matrix. We must also take into account that a high score in one query may not correlate to a better match than a lower score in another query. To adjust as best that we could we decided that we would normalize all scores from an individual query based on the highest score from that query. That way we would not have such large changes in values as we would get if we did not do this. After talking to Dr. Miguel and our faculty advisor Rana Bayrakçismith, we were told that we can know with a fairly high certainty that if images were taken within 90 seconds from each other that the snow leopard in that image are the same snow leopard. WIth this knowledge we decided that we would add to the recognition scores if the chips were taken from images within 90 seconds or if chips came from the same image.The image name we are provided with from Panthera has metadata that is explained in MCL/parsing ICD. We make sure however that by adding this we do not let the score go above 1 because we want the entire matrix to be normalized. The parameters for same set and same image can be changed under options -> edit preferences. Extensive testing still needs to be done to try to make HotSpotter as accurate as possible and go through with large databases which ECE 17.7 did not have time to do. Also it may be worth looking into not averaging scores and taking the higher or lower of two scores. Querying was not thoroughly tested enough so we cannot say if the scores entered are completely resembling what their actual matching strength relative to all other queries.

MCL:

Once we have a score matrix created from querying every chip we must then analyze all of the information we have created. We do this by using our score matrix as an undirected weighted graph and run a clustering algorithm on it. This means that we strengthen strong bonds and weaken weak bonds until we have disjoint clusters which represent individual snow leopards. In the MCL/parsing ICD we explain the process of MCL and how we create clusters. The main parameters that should be adjusted are the inflation parameter and the max loops parameter. The inflation parameter makes clusters tighter or looser. The lower the inflation number the fewer clusters the larger the number the more clusters. Max loops is the parameter that says the max amount of times that MCL is allowed to run. Running for more loops will ensure that the clusters will eventually become disjoint. These parameters need to be thoroughly tested and determined for the larger data sets that Panthera will be using. There are other disjoint clustering algorithms that may work better but we did not have time to develop and test other algorithms however. One of the requirements that Panthera had asked ECE 17.7 to accomplish that was unfeasible with time is creating folders of images that belong to individual snow leopards. This can be accomplished in the clusters to output function in mcl\_clustering.py where we already determine clusters.