The first step I took to determining cardinality was checking that games were repeated in the sales data.

SELECT name FROM vg\_csv GROUP BY name;

- 11493 row(s) returned

Since this is less than the 16598 rows in vg\_csv, I knew Video Game to Sales would be 1-M, the 1 on the Video Game side since there was no sales data repeated in the original vg\_csv.

Checking cardinality between Video Game and genre next,

SELECT name, genre FROM vg\_csv GROUP BY name, genre;

- 11493 row(s) returned

Because this is the same number as grouping only by name, I knew that every game has one genre. Checking the other way around doesn't even need a SELECT, I could immediately see repeated genres across different games. So I knew Genre to Video Game was 1-M.

I repeated the same steps for figuring out if every game has one Publisher. However, the number of rows returned when grouping by name and publisher was slightly bigger:

SELECT name, publisher FROM vg\_csv GROUP BY name, publisher;

- 11917 row(s) returned

This confused me, so I tried to investigate where the extra rows were coming from. At first I thought it might have something to do with the "N/A" publishers, but that was easy to disprove. I also decided to consider the "N/A" publishers missing data, rather than an optional relationship. I gave up on tracking down the additional rows, figuring the only thing that matter was whether the cardinality of Video Game to Publisher was one as I initially thought. I simply added a constraint for both game\_name and publisher\_id to be unique in my vg\_game table creation (I already had vgMigration.sql completed when I noticed the disparity in row counts).

CONSTRAINT test UNIQUE (game\_name, publisher\_id),

This didn't change the number of games in vg\_games, so I just carried on.

For Platform cardinality, I figured out early on that Video Games could be released on multiple platforms with separate sale data. As with genre, it was obvious by the small number of platforms that they would be repeated. However, I knew I would need a linking table between Platform and Video Game because I needed to preserve sale data between one game on multiple platforms.

The linking table I made was the Sales table, which tracked the individual releases as in the original vg\_csv. Video Game and Platform both have a 1-M relationship to Sales because there were instances where each were repeated throughout different rows in vg\_csv. For a while, I was just using game\_id and platform\_id as a composite primary key, but while normalizing the database, I saw that there were 5 duplicates. As it turned out, there were some games that were rereleased on the same platform, which I didn't think to look for. It was an easy fix, just needing to add a sale\_id. Since my Sales table most closely resembled vg\_csv, I put the rest of the attributes in there, leaving out global\_sales and ranking since I knew they could be calculated.

Tying Platform to Sales did create a problem with the addNewRelease procedure. Because they were linked, there wasn't really a way to include Platform in a new Video Game without creating sales (and year) data. The Platform parameter in addNewRelease was only there to insert into vg\_platform. I supposed that's a weakness with my design – it assumes that every game has sales data.