I would like to express my thoughts on moving our data to this flat file format, for I have concerns over whether this method will improve our organization's data management. First and foremost, I respect the decision and acknowledge the benefits of switching to this format. Flat file databases are easier to understand and require less skill to use when compared to the complexity of relational databases (Vines, 2011a). It also requires little effort to search for records using a flat file database because all information for that record is listed on the same row (Tuffill, 2015). For our current method of using relational databases, it can be difficult to search for information because it often requires joining two or more tables together. Yet though these benefits may appear useful for our organization, there are also several shortcomings to using this format. According to Doyle (2000), one of the issues with flat file databases is that you cannot add a new record into the database without entering its associated information for every single column. He states that this method results in records that repeatedly have the same values, thus leading to redundant data and wasted storage space. Furthermore, Doyle argues that the use of relational databases helps to resolve these issues and reduces both the amount of data entered and the storage space needed. Since our organization keeps track of hundreds or even thousands of flights, it is crucial that we utilize databases that require minimal storage space.

In addition to these points, there are several specific cases where this flat-file format can prove problematic for our organization. Vines (2011b) describes several different use cases regarding data entry and modification in a flat file database. One of these cases involves the insertion of data. Using our database as an example, it is impossible to add a new carrier name into our flat file without first adding a flight that involves a plane owned by that carrier. In addition, the author also claims that there are issues with removing and editing data. If you were to remove an airport from our flat file, you would also have to remove all entries involving that airport. Likewise, if you rename one of the carriers, you would have to rename that carrier for every single record in the database. This leads to another issue that the author describes—the possibility of inconsistent data. Vines (2011b) states that when a value is modified in a flat file database, that value does not change for all records and therefore it is possible for similar entries to have values inconsistent with one another. Finally, the author brings up the issue of redundant information within a flat file database. Every single time an airport code appears in our database, that airport's location, opening year, and number of terminals will also show up—even if that information is not needed at the time. In fact, our flat file also has redundant columns, since the values of the airport codes and airport origin codes are the same. This information becomes unnecessary and it wastes space within our database.

If our organization keeps its current relational database, then the issues stated above will be prevented because it will allow certain records to only appear once within the database. This helps to eliminate redundant or inconsistent information and it allows entries to be removed without the loss of data (Vines, 2011b). Therefore, I would strongly suggest that our organization preserves its current method for storing airline information. Though switching to a flat file format may seem easier at first, in the long run it is likely to make our operations more difficult and prone to error.

References

Doyle, S. (2000). *Understanding information technology.* Retrieved March 3, 2017, from https://books.google.com

Tuffill, S. (2015, March 31). Advantages & Disadvantages of Flat File Databases. Retrieved March 03, 2017, from https://www.techwalla.com/articles/advantages-disadvantages-of-flat-file-databases

Vines, R. (2011a, September 27). Databases from scratch II: Simple Database Design. Retrieved March 1, 2017, from http://www.geekgirls.com/2011/09/databases-from-scratch-ii-simple-database design/

Vines, R. (2011b, September 27). Databases from scratch III: The Design Process. Retrieved March 1, 2017, from http://www.geekgirls.com/2011/09/databases-from-scratch-iii-relational-design-process/