COMP1044 Coursework Assignment 2 Report A

Group 15 / Chew Language

Leong Chang Yung, XXXXXXXX
Lucas Dylan Purnell, XXXXXXXX
Tan Zhun Xian, XXXXXXXX Hao Wei, XXXXXXXX
Morhaf, XXXXXXXX
Thomas Tan Kean Yew, XXXXXXXX

April 1, 2021

1 Datatype Selection

For every field we chose the datatype depending on what would both match the data we had available and take up the lowest amount of memory possible while still ensuring that the type and length would be suitable for all new entries.

For most of the fields that were purely filled with numbers we set the type to either **int** or **decimal** (depending on whether the number was an integer or a decimal). A few examples of this are the *id* fields and the *length* and *rating* field in the *film* table. A notable exception to this was the *phone number* field. We set this datatype to **varchar** to allow people to enter their phone numbers in a different format despite all phone numbers in the dataset being integers. This was because many phone numbers need either a plus or a dash. Additionally, this field could be parsed and converted into an integer later if needed.

For all **varchar** we used the same principle but for the majority of fields we tried to keep the maximum lengths in neat values (normally multiples of 5). For instance, common values for the maximum length of **varchar** are 10, 25 and 50. We decided to set each field's maximum length to the longest value we thought would be realistically input. For instance, we set *movie_title* to **varchar(100)** as the longest name for a popular movie was 76 characters. We also set *first_name* and *last_name* to 50 as we thought that would be a good estimate on the longest name that anyone would realistically input.

All date values were set to the type **datetime** because they were all in the format YYYY-MM-DD hh:mm:ss. For instance the *last_update* and *create_date* fields.

2 Reasoning for Primary and Foreign Key Choice

When we were deciding on what to pick for the primary key for each time we understood that it had to be both unique and non-null. This is because the primary key serves as the identifier for each record in the table and if it is either repeated or NULL it can no longer serve as an identifier. The obvious candidate here was the ID field in every table(for instance the *inventory_id* in the inventory table) as it fulfils both of these criteria.

The foreign key has the same requirements. It needs to be both unique and independent in the array it belongs to. With the in mind we can use the same fields that we used as primary keys. Therefore for each of our arrays the *id* column is used to link the tables together. For instance, *customer_id* was chosen as the foreign key to link the *payment* table with the *customer* table and *address_id* was chosen as the foreign key to link the *staff* table with the *address* table.

3 Completeness of the Dataset

The database was relatively complete and we did not struggle with the quality of the data. We found that the data values were consistent across the different tables. This meant that we didn't run into any issues regarding the foriegn keys and we didn't find conflicting information in the data set.

We found a few different ways that the dataset was incomplete. The first was missing numbers from the auto increment data. For instance, there were missing *address_id*'s in *Address.csv*. This problem was easily remedied by setting it to **auto increment**. The next problem was that many fields that should not be NULL were NULL. For instance, some of the fields of *Rental_id* in *Payment.csv* were NULL when this should of not been the case. This is a problem as it may compromise the integrity of the database as this field being NULL will lead to orphaned records due to the lack of a foriegn key.

There was also a NULL in the *password* column of the *staff.csv* table. To remedy this we could of used the **notnull** function which would then prompt the user to provide an input for these NULL fields. However, we decided that it would be better to leave these fields blank to avoid sacrificing the integrity of the data.

4 Errors in the Database

We also noticed that in *country.csv* and *customer.csv* that there wasn't a consistent delimiter. The file alternated between using commas and semi-colons(to separate its values) depending on the column. This meant that it was impossible to import the data into a database while keeping it in its intended format. To remedy this we just changed the semi-colons for commas wherever applicable using a regular notepad application.

The other error that we found was that there was a corrupted picture file within *staff.csv*. This corruption it unable to view or display the aforementioned picture. We decided that having a picture within a staff database was not of high importance and wouldn't decrease the completeness or understandability of the dataset, so we just removed the picture. A possible solution was making the field type storing the picture a **longblob**. This allowed the picture to be imported, albeit in an incomplete state.