CS3101-PRACTICAL 2

DATE: 26TH NOVEMBER, 2018

STUDENT ID: 160016114

Design & Implementation:

1. Compilation, Execution & Usage Instruction:

To build the user interface for the database, I used PHP, HTML and CSS. I chose PHP as a comprehensive server-side solution, as all pages can be easily rendered on the server, which in a way prevents overloading of the client side. It is also extremely portable, i.e., can run on any server and every platform, ranging from Windows to Linux, which makes it easy for an individual to run the code in any environment. HTML is used to design the web page, while CSS is used to beautify the output given by the HTML code.

To access the interface, one can type the host name ‘gj25.host.cs.st-andrews.ac.uk’ on a web browser.

2. Overview:

This practical is primarily divided into four parts all of which revolve around interaction with a database made in mariadb in several ways. The requirement was to produce a database for an online streaming service for audiobooks. We were also required to populate the tables of the database that hold data regarding the audiobook itself, and people who have had any sort of interaction with the audiobook, which ranges from buying an audiobook, to narrating one, to being an author of one. Furthermore, we had to apply the right integrity constraints wherever required and also produce a user interface for a part of the system. All the basic requirements of this practical have been completed.

3. Database Implementation:

- **Creating MariaDB database and tables**

  Making the database and populating the tables in the database was a fairly simple task, and this was done primarily using the terminal. Based on the given ten schemas, ten tables were created and populated in the database, with the same names and attributes as that of the schemas. For populating the database, two methods were used. The first method was converting the sample data, which is given in xlsx format, to csv

format, and then using the ‘LOAD DATA INFILE’ statement, which reads rows from the csv file into a table at an extremely high speed. For populating tables with a large number of attributes, such as person, publisher, audiobook and chapter, the aforementioned command was extremely useful, as it saved the tedious task of entering data in each row separately.

For the remaining tables, I added the data by using the ‘INSERT INTO TABLE’, statement, as there were some attributes whose types were not being recognised correctly by the former statement, and hence wrong data was being entered into the tables.

As there was no ID given in the sample data, an ID attribute was made as per the given schema, since it acts as a primary key to the person table and also a foreign key to many other tables in the database. No integrity constraints were applied while populating the tables at first. It was only after all the tables were complete with their respective data, is when the the ‘ALTER TABLE’ statement was used to define primary keys and foreign keys. The fields whose values weren’t given in the sample data, like audiofile, which is a part of the audiobook table, were given the value NULL. The scenario is similar in all the tables, where certain fields are empty; they have just been assigned a null value, apart from the table phone_number, where phone_number field is left blank for customers without a phone number since phone_number is a primary key of the table phone_number and a primary key can never be NULL.

- **SQL QUERIES**

The next step was to specify the queries given in the specification in SQL.

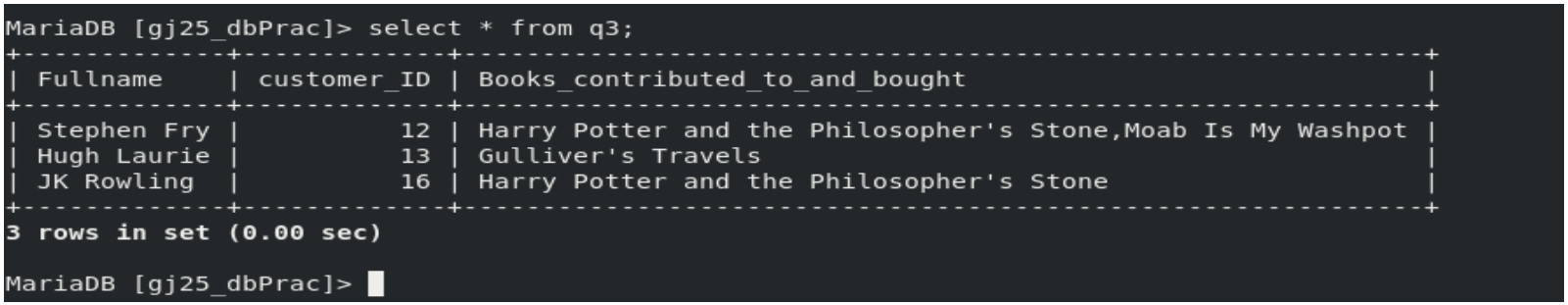
The first query prints a list of all customers whether or not they have bought any books. The query prints out a single row per customer, with some attributes being straightforward like customer ID, email address, while the attribute full name was gotten by concatenating the three fields(forename, middle_initals, surname) of the person table. As if one can observer from the sample data, not every person has middle_initals, so in order to prevent extra spaces in between first name and last name, an ‘IFNULL’ condition was implemented before displaying the name, which checks that if the value of middle_initial is null, then concatenate only the forename and the surname fields. We also need to print the total audiobooks purchased by each customer, which is done by counting the number of purchase dates of a customer and to print the the total money spent by a customer, which is calculated by getting the sum of the purchase prices of each customer. If a customer hasn’t bought any books, the purchase price is displayed as “0.00”. This data is mainly taken from the person table, which is in a natural join relation with the customer table, to get only the customers and not any of the people who narrate or author audiobooks. The person table is further in a left join relation with audiobook_purchases, and the audiobook_purchases is further in a left join relation with audiobook, so that the total number of audiobooks purchased by the customer can be received. This received value is then put in the sum aggregate, and the whole query is grouped by the person.ID, so that there exists only row per customer. This SQL query is put in a view called ‘q1’ which is a part of the database, and can be called through the ‘select’ statement.

```
MariaDB [gj25_dbPrac]> select * from q1;
+----------------------+-----------------------+-----------+-------------------------+--------------------+
| fullname             | email_address         | person_ID | total_books_purchased | total_money_spent |
+----------------------+-----------------------+-----------+-------------------------+--------------------+
| Bob B A Bobson       | bob_jnr@bobson.com    |        10 |                       0 | 0.00               |
| Bob A B Bobson       | bob_snr@bobson.com    |        11 |                       0 | 0.00               |
| Stephen Fry          | sfry@email.com        |        12 |                       2 | 29.19              |
| Hugh Laurie          | hugh@laurie.com       |        13 |                       1 | 38.00              |
| Ruth Letham          | ruth@letham.com       |        14 |                       1 | 38.00              |
| Simon Prebble        | NULL                  |        15 |                       0 | 0.00               |
| JK Rowling           | jk@rowling.com        |        16 |                       2 | 45.19              |
| Newton A F Scamander | NULL                  |        17 |                       0 | 0.00               |
| Pippa A Smith        | pippa.smith@email.com |        18 |                       1 | 16.00              |
| Jon Q Spellbad       | jon@spellbad.com      |        19 |                       0 | 0.00               |
| Jonathan Swift       | NULL                  |        20 |                       0 | 0.00               |
+----------------------+-----------------------+-----------+-------------------------+--------------------+
11 rows in set (0.00 sec)
```

For the second SQL query, to display the number of audibooks that have not been purchased by anyone, the ISBN and title is displayed from the audiobook table, where audiobook is in a relation with audiobook_purchases, so that only the rows of the books are chosen audiobook, that are not in audiobook_purchases. There is only one book which isn't purchased by anyone, so the condition of ordering books in ascending order doesn't hold any significance, unless there are other books that have not been bought. This SQL query is put in a view called 'q2' which is a part of the database, and can be called through the 'select' statement.

```
MariaDB [gj25_dbPrac]>
MariaDB [gj25_dbPrac]> Select audiobook.ISBN, audiobook.title
    -> from audiobook
    -> left join audiobook_purchases
    -> on  audiobook_purchases.ISBN = audiobook.ISBN
    -> where audiobook_purchases.ISBN is null
    -> order by title asc;
+----------------+-----------------------------------------+
| ISBN           | title                                   |
+----------------+-----------------------------------------+
| 860-1404211171 | Fantastic Beasts and Where to Find Them |
+----------------+-----------------------------------------+
1 row in set (0.01 sec)

MariaDB [gj25_dbPrac]> select * from q2;
+----------------+-----------------------------------------+
| ISBN           | title                                   |
+----------------+-----------------------------------------+
| 860-1404211171 | Fantastic Beasts and Where to Find Them |
+----------------+-----------------------------------------+
1 row in set (0.00 sec)

MariaDB [gj25_dbPrac]>
```

The third query asks us to list all the contributors who have bought audiobooks they authored and/or narrated. The query is supposed to print out contributor_ID which is taken from audiobook_purchases along with fullname which is printed in the same way that is describe for query 1. It also displays title of all the audiobooks bought by people who have authored and/or narrated the book, displayed in comma seperated list format. The first left join between audiobook_purchases and audiobook_authors, gets all the people who have bought an audiobook they authored, while the second join states the people who have bought the audiobooks they authored and/or narrated. This is finally joined by person to get the names of people who satisfy the aforementioned condition. The output is ordered by the ascending order of customer ID, and the books in the comma separated list appear using the group_concat statement on audiobook.title.
This SQL query is put in a view called ‘q3’, which is also a part of the database and can be called through the select statement.



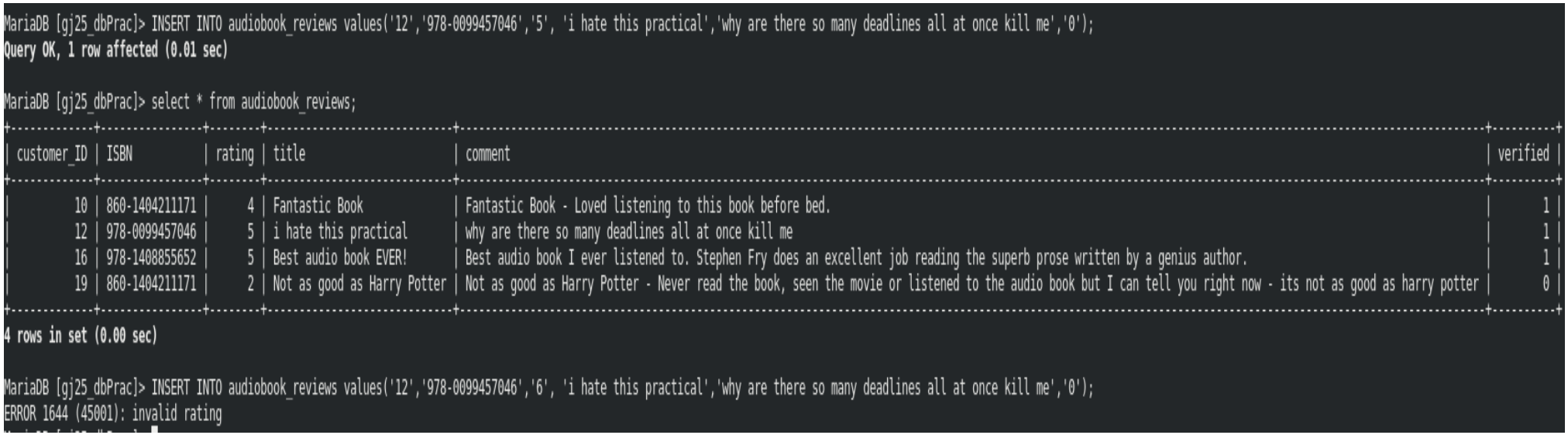
```
MariaDB [gj25_dbPrac]> select * from q3;
+-------------+-------------+-----------------------------------------------------------+
| Fullname    | customer_ID | Books_contributed_to_and_bought                           |
+-------------+-------------+-----------------------------------------------------------+
| Stephen Fry |          12 | Harry Potter and the Philosopher's Stone,Moab Is My Washpot |
| Hugh Laurie |          13 | Gulliver's Travels                                        |
| JK Rowling  |          16 | Harry Potter and the Philosopher's Stone                  |
+-------------+-------------+-----------------------------------------------------------+
3 rows in set (0.00 sec)

MariaDB [gj25_dbPrac]>
```

```
MariaDB [gj25_dbPrac]> SELECT
    -> ifnull(concat(person.forename, " ", person.middle_initials," ", person.surname), concat(person.forename, " ", person.surname))
    ->  as Fullname, audiobook_purchases.customer_ID, group_concat(audiobook.title ORDER BY audiobook.title asc) as Books_contributed_to_and_bought
    ->  from audiobook_purchases
    ->  left join audiobook_authors
    ->  on audiobook_purchases.customer_ID = audiobook_authors.contributor_ID AND audiobook_purchases.ISBN = audiobook_authors.ISBN
    ->  join audiobook
    ->  on (audiobook_purchases.ISBN = audiobook.ISBN AND audiobook_purchases.customer_ID = audiobook.narrator_ID) OR audiobook.ISBN = audiobook_authors.ISBN
    ->  join person
    ->  on audiobook_purchases.customer_ID = person.ID
    ->  group by customer_ID ORDER BY customer_id asc;
+-------------+-------------+-----------------------------------------------------------+
| Fullname    | customer_ID | Books_contributed_to_and_bought                           |
+-------------+-------------+-----------------------------------------------------------+
| Stephen Fry |          12 | Harry Potter and the Philosopher's Stone,Moab Is My Washpot |
| Hugh Laurie |          13 | Gulliver's Travels                                        |
| JK Rowling  |          16 | Harry Potter and the Philosopher's Stone                  |
+-------------+-------------+-----------------------------------------------------------+
3 rows in set (0.00 sec)
```

- **TRIGGERS & PROCEDURES**
  Since we were given a choice of implementing either triggers or procedures in the first two parts of question three, I chose to implement triggers for both parts.

For part 1

There is a trigger called ‘check_audiobook_ratings’, which has a variable called personid, that holds that details of a person, who may or may not have purchased a book through this system. This variable is further passed in an if statement to verify the latter, and if the person has bought the book from this system, then verified is marked as true or ‘1’ in this case since the type of verified is tinyint. The trigger also throws an error if this person’s audiobook rating is not within the scale of 1 to 5. This trigger executes before any values can be inserted into the table audiobook_reviews.



```
MariaDB [gj25_dbPrac]> INSERT INTO audiobook_reviews values('12','978-0099457046','5', 'i hate this practical','why are there so many deadlines all at once kill me','0');
Query OK, 1 row affected (0.01 sec)

MariaDB [gj25_dbPrac]> select * from audiobook_reviews;
+-------------+---------------+--------+--------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+----------+
| customer_ID | ISBN          | rating | title                    | comment                                                                                                                                                                           | verified |
+-------------+---------------+--------+--------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+----------+
|          10 | 860-1404211171 |      4 | Fantastic Book           | Fantastic Book - Loved listening to this book before bed.                                                                                                                         |        1 |
|          12 | 978-0099457046 |      5 | i hate this practical    | why are there so many deadlines all at once kill me                                                                                                                               |        1 |
|          16 | 978-1408855652 |      5 | Best audio book EVER!    | Best audio book I ever listened to. Stephen Fry does an excellent job reading the superb prose written by a genius author.                                                       |        1 |
|          19 | 860-1404211171 |      2 | Not as good as Harry Potter | Not as good as Harry Potter - Never read the book, seen the movie or listened to the audio book but I can tell you right now - its not as good as harry potter |        0 |
+-------------+---------------+--------+--------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+----------+
4 rows in set (0.00 sec)

MariaDB [gj25_dbPrac]> INSERT INTO audiobook_reviews values('12','978-0099457046','6', 'i hate this practical','why are there so many deadlines all at once kill me','0');
ERROR 1644 (45001): invalid rating
```

For part 2

There is a trigger called ‘verify_age’, in which there exists two variables, one being age and another one being ageRating. To check if the customer is too young to purchase a book, the date_of_birth from the table person is taken, and it is put in the timestampdiff function, where the current date is also taken as a parameter, this then returns a value in years by subtracting the date of birth of the person by the current date.

The second variable ageRating takes age rating for a specific audiobook which the customer is trying to buy.

Finally, there’s a comparison to check if the age is less than age rating; if this turns out to be true, then an error is thrown preventing the customer from buying a book they are too young to purchase. This trigger is executed before any values can be inserted into the audiobook_purchases table.

```
MariaDB [gj25_dbPrac]> select * from audiobook_purchases;
+-------------+----------------+---------------------+
| customer_ID | ISBN           | purchase_date       |
+-------------+----------------+---------------------+
|          12 | 978-0099457046 | 2018-10-23 21:34:02 |
|          12 | 978-1408855652 | 2018-10-23 21:29:48 |
|          13 | 978-0393957242 | 2018-10-23 21:30:10 |
|          14 | 978-0393957242 | 2018-10-23 21:35:54 |
|          16 | 978-0393957242 | 2018-10-23 21:30:34 |
|          16 | 978-1408855652 | 2018-10-23 21:31:08 |
|          18 | 978-1611749731 | 2018-10-23 21:30:47 |
+-------------+----------------+---------------------+
7 rows in set (0.00 sec)

MariaDB [gj25_dbPrac]> insert into audiobook_purchases values('10', '978-1611749731','2018-10-23 21:34:02');
ERROR 1644 (45001): Too young to purchase the book
MariaDB [gj25_dbPrac]> insert into audiobook_purchases values('12', '978-1611749731','2018-10-23 21:34:02');
Query OK, 1 row affected (0.00 sec)

MariaDB [gj25_dbPrac]> select * from audiobook_purchases;
+-------------+----------------+---------------------+
| customer_ID | ISBN           | purchase_date       |
+-------------+----------------+---------------------+
|          12 | 978-0099457046 | 2018-10-23 21:34:02 |
|          12 | 978-1408855652 | 2018-10-23 21:29:48 |
|          12 | 978-1611749731 | 2018-10-23 21:34:02 |
|          13 | 978-0393957242 | 2018-10-23 21:30:10 |
|          14 | 978-0393957242 | 2018-10-23 21:35:54 |
|          16 | 978-0393957242 | 2018-10-23 21:30:34 |
|          16 | 978-1408855652 | 2018-10-23 21:31:08 |
|          18 | 978-1611749731 | 2018-10-23 21:30:47 |
+-------------+----------------+---------------------+
8 rows in set (0.00 sec)
```

For part 3

Since the students aren’t given the necessary permissions to revoke access of on the school systems, I have described how one would go about revoking direct insertion into the person relation.

One can revoke direct entry into the person relation by using the revoke statement.

There can be two situations where the authorisation to directly insert into the person relation would be revoked for only one user, or for multiple users. Hypothetically assuming for there to be 3 users U1, U2 and U3. U3 is being granted privileges by U1.

if revoke statement has a cascade in the end, then the procedure would include a revoke statement which would look something like

REVOKE INSERT ON person from U1,U2 cascade

This would revoke direct entry on to the person relation for users U1, U2 and U3 as well since using cascade at the end of the revoke statement also invalidates the privileges from anyone to whom U1 had granted privileges to.

However if revoke statement has a restrict in the end, then the procedure would include a revoke statement which would look something like

REVOKE INSERT ON person from U1,U2 restrict

This would revoke direct entry on to the person relation for users U1 but not U2. Instead it will return an error code as the U2 has passed on the specified privileges to another user, and hence there is a cascade required.

There are two procedures that are made as stated in the practical specification, insertCustomer, which is supposed to add new values into the person table and the customer table, and insertContributor, which is supposed to add new values into the person table and the contributor table. In the insertCustomer procedure, there is an 'if' statement which basically checks if the email address in the customer table is being added in the right format, i.e., if there is an '@' sign existing in the email address at the right position. If it doesn't exist, then the procedure throws an error and prevents insertion into both tables.

4. GUI Implementation:

The web interface displays a list of audiobooks in a dropdown menu, and shows the review for any of the listed audiobooks, if the reviews exist.

For building my GUI, I used PHP, HTML and CSS. The Implementation of my code is inspired from the php tutorial of w3schools.com. There are 2 PHP files, which include also implement HTML and CSS code in them. The index.php is used to connect to the database, where it throws an error should the connection be unsuccessful. After making the connection to the database, it performs an sql query against the connected database to get the title and ISBN from the audiobook table. These values are then used in the dropdown menu to display the list of audiobooks to review by putting using the ISBN as an index to recognize the names of the audiobooks and display them using the while loop. Finally in the script tag, function showUSR is implemented which creates an XMLHttpRequest object for communicating with the server. An XMLHttpRequest object called xmlhttp is created which calls function() whenever the readyState property changes. The readyState property holds the status of the specific location of the data when it is being transferred between a web browser and a web server. So if the said property is 4 and the status of xmlhttp is 200, that mean that the response is ready and the html version of the element "txtHint" is changed with the response given from the web server, which is stored in xmlhttp.responseText".

This file also contains CSS that is used to beautify the web interface built using HTML. The reviews of audiobooks are being displayed in a table, where the colour of the table data rows change if the cursor hovers upon the table. The background of the webpage is an image taken from the web.

The main purpose of the second file, which is the phpscript.php file is to be called in the index.php file so that it can make the appropriate tables on the web page using HTML, and add the appropriate values in the tables.

Word Count: 2183