Islington college
(इस्लिङ्टन कलेज)

LONDON METROPOLITAN UNIVERSITY

Module Code & Module Title

CC4057NI Introduction to Information Systems

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: Bijay Bharati

Group: L1C4

College ID: NP01CP4A190041

Assignment Due Date: 20th December 2019

Assignment Submission Date: 20th December 2019

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Contents

1 INTRODUCTION 1
1.1 Databases 1
1.2 Description of organization 2
1.3 Description of project 2
2 Database Model 3
2.1 Classical ERD 3
2.2 Relational Diagram 4
3 Tables 5
3.1 Users 5
3.1.1 Explanation 5
3.1.2 Screenshot of table creation 5
3.1.3 Description of table 6
3.1.4 Screenshot of data insertion 6
3.1.5 Selecting all inserted data 6
3.2 Genres 7
3.2.1 Explanation 7
3.2.2 Screenshot of table creation 7
3.2.3 Description of table 7
3.2.4 Screenshot of data insertion 8
3.2.5 Selecting all inserted data 9
3.3 Publishers 10
3.3.1 Explanation 10
3.3.2 Screenshot of table creation 10
3.3.3 Description of table 10
3.3.4 Screenshot of data insertion 11
3.3.5 Selecting all inserted data 11
3.4 Games 12
3.4.1 Explanation 12
3.4.2 Screenshot of table creation 12
3.4.3 Description of table 13
3.4.4 Screenshot of data insertion 13

3.4.5 Selecting all inserted data 13
3.5 Purchases 14
3.5.1 Explanation 14
3.5.2 Screenshot of table creation 14
3.5.3 Description of table 14
3.5.4 Screenshot of data insertion 15
3.5.5 Selecting all inserted data 15
3.6 OrderedGames 16
3.6.1 Explanation 16
3.6.2 Screenshot of table creation 16
3.6.3 Description of table 16
3.6.4 Screenshot of data insertion 17
3.6.5 Selecting all inserted data 17
4 Data Dictionary 18
4.1 Data Dictionary for Users 18
4.2 Data Dictionary for Genres 18
4.3 Data Dictionary for Publishers 18
4.4 Data Dictionary for Games 19
4.5 Data Dictionary for Purchases 19
4.6 Data Dictionary for OrderedGames 20
5 Queries 21
Example of different queries 21
5.1 BETWEEN 21
5.2 ORDER BY 21
5.3 IN 22
5.4 LIKE 22
5.5 LIMIT 23
5.6 COUNT 23
5.7 GROUP BY 24
5.8 HAVING 25
5.9 DISTINCT 26
5.10 JOIN 27

5.10.1 INNER JOIN 28
5.10.2 LEFT JOIN 29
5.10.3 RIGHT JOIN 31
5.10.4 OUTER JOIN 33
6 CONCLUSION 34
6.1 Research and Conclusion 34
Bibliography 35

Figures

Figure 1: Classical ERD of an online game distribution service 3
Figure 2: Relational Diagram 4
Figure 3: Creating Users Table 5
Figure 4: Description of Users 6
Figure 5: Inserting data into Users 6
Figure 6: Selecting all data from Users 6
Figure 7: Creating Genres table 7
Figure 8: Description of Genres 7
Figure 9: Inserting data into Genres 8
Figure 10: Selecting all data from Genres 9
Figure 11: Creating Publishers table 10
Figure 12: Description of Publishers 10
Figure 13: Inserting data into Publishers 11
Figure 14: Selecting all data from Publishers 11
Figure 15: Creating Games table 12
Figure 16: Description of games 13
Figure 17: Inserting data into Games 13
Figure 18: Selecting all data from games 13
Figure 19: Creating Purchases table 14
Figure 20: Description of Purchases 14
Figure 21: Inserting data into Purchases 15
Figure 22: Selecting all data from Purchases 15
Figure 23: Creating OrderedGames table 16
Figure 24: Description of OrderedGames 16
Figure 25: Inserting data into OrderedGames 17
Figure 26: Selecting all data from OrderedGames 17
Figure 27: BETWEEN 21
Figure 28: ORDER BY 21
Figure 29: IN 22
Figure 30: Like 22
Figure 31: LIMIT 23
Figure 32: COUNT 23
Figure 33: Data from Games 24
Figure 34: GROUP BY 24
Figure 35: Data from Games 25
Figure 36: Grouped data 25
Figure 37: HAVING 25
Figure 38: Data from Purchases 26
Figure 39: DISTINCT 26
Figure 40: Joining all data from Games and Publishers 28

Figure 41: INNER JOIN 28
Figure 42: Data from Publishers 29
Figure 43: Data from Games 29
Figure 44: LEFT JOIN 30
Figure 45: Data from Publishers 31
Figure 46: Data from Games 31
Figure 47: RIGHT JOIN 32
Figure 48: OUTER JOIN 33

1 INTRODUCTION

1.1 Databases

Data can be considered as information, especially facts or numbers, collected to be examined and considered and used to help decision-making, or information in an electronic form that can be stored and used by a computer:

The data was/were collected by various researchers.

Now the data is/are being transferred from magnetic tape to hard disk. (©Cambridge University Press, 2019)

Database is a collection of data or information organized for rapid search and retrieval, especially by a computer. Databases are structured to facilitate storage, retrieval, modification and deletion of data in conjunction with various data-processing operations. (ENCYCLOPEDIA Britannica©, 2013).

The software which is used to manage database is called Database Management System (DBMS). For Example, MySQL, Oracle etc. are popular commercial DBMS used in different applications. DBMS allows users the following tasks:

Data Definition: It helps in creation, modification and removal of definitions that define the organization of data in database.

Data Updating: It helps in insertion, modification and deletion of the actual data in the database.

Data Retrieval: It helps in retrieval of data from the database which can be used by applications for various purposes.

User Administration: It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control and recovering information corrupted by unexpected failure. (GeeksforGeeks, 2019)

Every organization uses a database. No matter how small or big a business is, it still needs to keep records of its activities, this is where a database steps in. A database is much more reliable, secure, manageable and flexible compared to traditional methods of storing vital information in a physical file. Database based system almost replaced the traditional method of storing information in paper files. Many people and even government bodies are switching to using a database and rapidly upgrading their office environment.

Databases (or DBs) have played a very important part in the recent evolution of computers. The first computer programs were developed in the early 1950s, and focused almost completely on coding languages and algorithms. At the time, computers were basically giant calculators and data (names, phone numbers) was considered the

leftovers of processing information. Computers were just starting to become commercially available, and when business people started using them for real-world purposes, this leftover data suddenly became important. (Foote, 2017)

It is clear that databases were created for making it easier for people and businesses to keep record of their findings and utilize the data for their benefit.

1.2 Description of organization

It has been discussed why an organization or a business needs a database, now let's consider any type of organization and discuss about it.

Say, there is a Digital Distribution Service for video games which just means it is a platform (consider it as an app like play store or steam). Now this service is managed and owned by a company so there will be customers i.e. the users. Games are bought and managed by users through this service so it can be considered as a digital store for users to buy all their games and manage their game library.

A database is used to keep track of all the users who use the service, purchases made, games bought, etc. There are many more features that a service like this could have like providing video streaming services, social networking services and matchmaking services all through a single platform but in this representation those are not included. This is just meant to be a simple representation to buy games.

1.3 Description of project

The project was about creating a database and being able to explain it aiming to make students familiar with queries, data dictionary, relational diagrams etc. This particular project is about a database for a Digital Distribution Service for video games. My objective through the completion of this project is to improve my own knowledge about databases and gain better understanding about how a database is implemented, how the different entities in a database are related and to be confident in creating and using and managing a database.

The database contains 6 entities (Users, Purchases, OrderedGames, Games, Genres and Publishers).

Each entity has attributes which are the properties of an entity that further describe what it is. The details are mentioned in the upcoming section "Database Model" to prevent repetition.

2 Database Model

2.1 Classical ERD

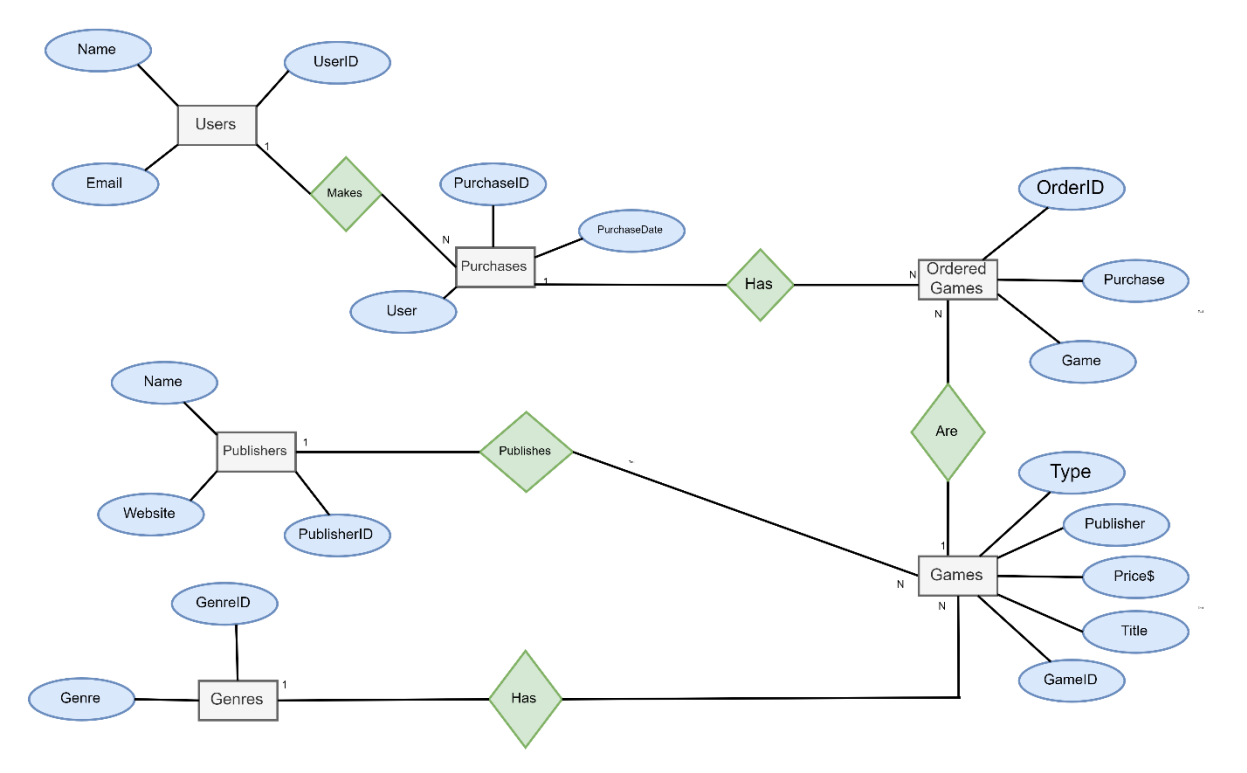


Figure 1: Classical ERD of an online game distribution service

2.2 Relational Diagram

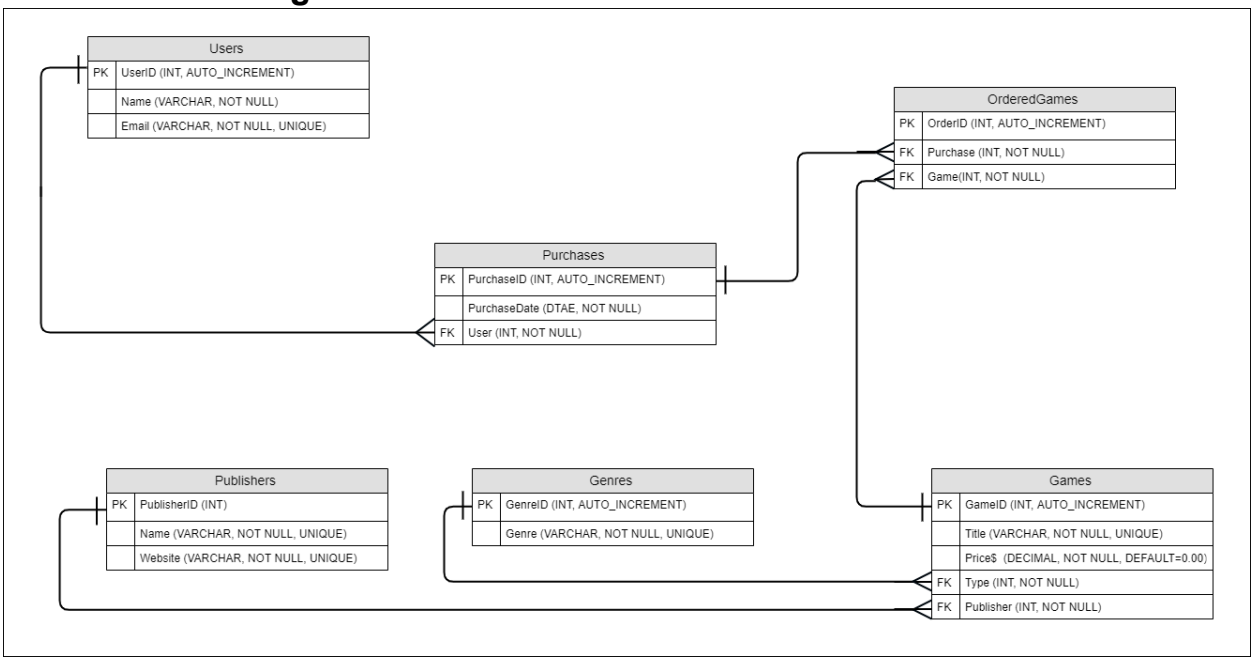


Figure 2: Relational Diagram

3 Tables

Entity when put in a database is a table. The attributes become the column and values are inserted.

3.1 Users

3.1.1 Explanation

This table represents the users of the service in a sense, they are the customers. Users has 3 attributes:

- UserID
- Name
- Email

UserID is primary key, has INT data type and is used to identify everyone individually.

Name is the name of the user, has VARCHAR data type and is NOT NULL because a user has a name.

Email is the email of the user, has VARCHAR data type because emails have various characters and is UNIQUE and NOT NULL as, users need to provide an email to contact them and 2 users cannot use the same email.

3.1.2 Screenshot of table creation

```
Setting environment for using XAMPP for Windows.
B.B@B b:\programs\xampp
# mysql -u root -h localhost
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.8-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE OnlineGameDistributionService;
Query OK, 1 row affected (0.491 sec)

MariaDB [(none)]> USE OnlineGameDistributionService;
Database changed
MariaDB [OnlineGameDistributionService]> CREATE TABLE Users(UserID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(50) NOT NULL, Email
 VARCHAR(255) NOT NULL UNIQUE);
Query OK, 0 rows affected (0.539 sec)

MariaDB [OnlineGameDistributionService]>
```

Figure 3: Creating Users Table

3.1.3 Description of table

```
MariaDB [OnlineGameDistributionService]> DESCRIBE Users;
+--------+--------------+------+-----+---------+----------------+
| Field  | Type         | Null | Key | Default | Extra          |
+--------+--------------+------+-----+---------+----------------+
| UserID | int(11)      | NO   | PRI | NULL    | auto_increment |
| Name   | varchar(50)  | NO   |     | NULL    |                |
| Email  | varchar(255) | NO   | UNI | NULL    |                |
+--------+--------------+------+-----+---------+----------------+
3 rows in set (0.184 sec)
```

Figure 4: Description of Users

3.1.4 Screenshot of data insertion

```
MariaDB [OnlineGameDistributionService]> INSERT INTO Users VALUES(1000,"Reigen","reigen@mail.com"),
    -> ("","Mob","mob@mail.com"),
    -> ("","Saitama","saitama@mail.com"),
    -> ("","Genos","genos@mail.com"),
    -> ("","Dimple","dimple@mail.com");
Query OK, 5 rows affected, 4 warnings (0.128 sec)
Records: 5  Duplicates: 0  Warnings: 4

MariaDB [OnlineGameDistributionService]>
```

Figure 5: Inserting data into Users

3.1.5 Selecting all inserted data

```
MariaDB [OnlineGameDistributionService]> SELECT*FROM Users;
+--------+---------+------------------+
| UserID | Name    | Email            |
+--------+---------+------------------+
|   1000 | Reigen  | reigen@mail.com  |
|   1001 | Mob     | mob@mail.com     |
|   1002 | Saitama | saitama@mail.com |
|   1003 | Genos   | genos@mail.com   |
|   1004 | Dimple  | dimple@mail.com  |
+--------+---------+------------------+
5 rows in set (0.000 sec)
```

Figure 6: Selecting all data from Users

3.2 Genres

3.2.1 Explanation

This table represents the different game genres. This table was created because there are thousands of games and if somebody wants to play a game like counter strike for example then he has to look up the shooting genre. Genres has two attributes:

- GenreID
- Genre

GenreID is primary key, has INT data type and is auto incremented, this makes it easy to update the table and add new genre or identify a genre.

Genre is just the name, it has VARCHAR data type, is NOT NULL and UNIQUE because a category i.e. a genre needs to be unique and cannot be empty.

3.2.2 Screenshot of table creation

```
MariaDB [OnlineGameDistributionService]> CREATE TABLE Genres(GenreID INT PRIMARY KEY AUTO_INCREMENT,
Genre VARCHAR(255) UNIQUE NOT NULL);
Query OK, 0 rows affected (0.277 sec)

MariaDB [OnlineGameDistributionService]>
```

Figure 7: Creating Genres table

3.2.3 Description of table

```
MariaDB [OnlineGameDistributionService]> DESCRIBE Genres;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| GenreID | int(11)      | NO   | PRI | NULL    | auto_increment |
| Genre   | varchar(255) | NO   | UNI | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
2 rows in set (0.042 sec)
```

Figure 8: Description of Genres

3.2.4 Screenshot of data insertion

```
MariaDB [OnlineGameDistributionService]> INSERT INTO Genres VALUES(1,"Action"),
    -> ("","Platformer"),
    -> ("","Shooter"),
    -> ("","Fighting"),
    -> ("","Beat'em up"),
    -> ("","Stealth"),
    -> ("","Survival"),
    -> ("","Battle Royale"),
    -> ("","Rhythm"),
    -> ("","Action Adventure"),
    -> ("","Survival Horror"),
    -> ("","Metroidvania"),
    -> ("","Adventure"),
    -> ("","Text Adventure"),
    -> ("","Graphic adventure"),
    -> ("","Visual novels"),
    -> ("","Interactive movie"),
    -> ("","RT 3-D adventure"),
    -> ("","Roleplaying"),
    -> ("","ARPG"),
    -> ("","MMORPG"),
    -> ("","Rougelike"),
    -> ("","TaticalRPG"),
    -> ("","SandboxRPG"),
    -> ("","First Person Party Based RPG"),
    -> ("","Simulation"),
    -> ("","Construction and management sim"),
    -> ("","Life simulation"),
    -> ("","Vehicle simulation"),
    -> ("","Strategy"),
    -> ("","4x"),
    -> ("","Artillery"),
    -> ("","Autochess"),
    -> ("","MOBA"),
    -> ("","RTS"),
    -> ("","RTT"),
    -> ("","TD"),
    -> ("","TBS"),
    -> ("","TBT"),
    -> ("","Wargame"),
    -> ("","Grand Strategy Wargame"),
    -> ("","Sports"),
    -> ("","Casual"),
    -> ("","Logic"),
    -> ("","Racing"),
    -> ("","Competetive"),
    -> ("","Sports-based fighting"),
    -> ("","Board Card game"),
    -> ("","DCCG"),
    -> ("","MMO");
Query OK, 50 rows affected, 49 warnings (0.142 sec)
Records: 50  Duplicates: 0  Warnings: 49
```

Figure 9: Inserting data into Genres

3.2.5 Selecting all inserted data

```
MariaDB [OnlineGameDistributionService]> SELECT*FROM Genres ORDER BY GenreID;
+---------+---------------------------------+
| GenreID | Genre                           |
+---------+---------------------------------+
|       1 | Action                          |
|       2 | Platformer                      |
|       3 | Shooter                         |
|       4 | Fighting                        |
|       5 | Beat'em up                      |
|       6 | Stealth                         |
|       7 | Survival                        |
|       8 | Battle Royale                   |
|       9 | Rhythm                          |
|      10 | Action Adventure                |
|      11 | Survival Horror                 |
|      12 | Metroidvania                    |
|      13 | Adventure                       |
|      14 | Text Adventure                  |
|      15 | Graphic adventure               |
|      16 | Visual novels                   |
|      17 | Interactive movie               |
|      18 | RT 3-D adventure                |
|      19 | Roleplaying                     |
|      20 | ARPG                            |
|      21 | MMORPG                          |
|      22 | Rougelike                       |
|      23 | TaticalRPG                      |
|      24 | SandboxRPG                      |
|      25 | First Person Party Based RPG    |
|      26 | Simulation                      |
|      27 | Construction and management sim |
|      28 | Life simulation                 |
|      29 | Vehicle simulation              |
|      30 | Strategy                        |
|      31 | 4x                              |
|      32 | Artillery                       |
|      33 | Autochess                       |
|      34 | MOBA                            |
|      35 | RTS                             |
|      36 | RTT                             |
|      37 | TD                              |
|      38 | TBS                             |
|      39 | TBT                             |
|      40 | Wargame                         |
|      41 | Grand Strategy Wargame          |
|      42 | Sports                          |
|      43 | Casual                          |
|      44 | Logic                           |
|      45 | Racing                          |
|      46 | Competetive                     |
|      47 | Sports-based fighting           |
|      48 | Board Card game                 |
|      49 | DCCG                            |
|      50 | MMO                             |
+---------+---------------------------------+
50 rows in set (0.001 sec)
```

Figure 10: Selecting all data from Genres

3.3 Publishers

3.3.1 Explanation

This table contains information about the publishers. This table was created so that we can know the companies that publish a game. Publisher was used instead of developer because a publisher finances the game's development and marketing, usually the developer and the publisher are the same but in cases where they are not same, the publisher makes the game available. Following attributes are used:

- PublisherID
- Name
- Website

PublisherID is primary key and INT data type is used. PublisherID identifies the publisher. INT data type is used to make it easier for referencing as it is faster than typing the name of the publisher.

Name is the name of the publisher, VARCHAR data type is used and it is UNIQUE and NOT NULL. A publisher needs a unique name and the value cannot be null.

Website is the website of the publisher where all the information about contact, support etc. can be found. VARCHAR datatype is used because a website has multiple characters. It is also NOT NULL and UNIQUE as, every publisher has a website and no two publishers have the same website.

3.3.2 Screenshot of table creation

```
MariaDB [OnlineGameDistributionService]> CREATE TABLE Publishers(PublisherID INT PRIMARY KEY,
Name VARCHAR(255) UNIQUE NOT NULL, Website VARCHAR(255) UNIQUE NOT NULL);
Query OK, 0 rows affected (0.311 sec)
```

Figure 11: Creating Publishers table

3.3.3 Description of table

```
MariaDB [OnlineGameDistributionService]> DESCRIBE Publishers;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| PublisherID | int(11)      | NO   | PRI | NULL    |       |
| Name        | varchar(255) | NO   | UNI | NULL    |       |
| Website     | varchar(255) | NO   | UNI | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.018 sec)
```

Figure 12: Description of Publishers

3.3.4 Screenshot of data insertion

```
MariaDB [OnlineGameDistributionService]> INSERT INTO Publishers VALUES(1902,"Valve","valvesoftware.com"),
    -> (3648,"Blizzard Entertainment","blizzard.com"),
    -> (7777,"Mojang","mojang.com"),
    -> (8969,"CD Projekt","cdprojekt.com"),
    -> (1222,"Riot Games","riotgames.com");
Query OK, 5 rows affected (0.289 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Figure 13: Inserting data into Publishers

3.3.5 Selecting all inserted data

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Publishers;
+-------------+------------------------+--------------------+
| PublisherID | Name                   | Website            |
+-------------+------------------------+--------------------+
|        1222 | Riot Games             | riotgames.com      |
|        1902 | Valve                  | valvesoftware.com  |
|        3648 | Blizzard Entertainment | blizzard.com       |
|        7777 | Mojang                 | mojang.com         |
|        8969 | CD Projekt             | cdprojekt.com      |
+-------------+------------------------+--------------------+
5 rows in set (0.116 sec)
```

Figure 14: Selecting all data from Publishers

3.4 Games

3.4.1 Explanation

This table contains the information on games. The database is about a game distribution service so making a table about games is quite self-explanatory. Following are the attributes:

- GameID
- Title
- Price$
- Type
- Publisher

GameID is primary key and it has INT data type. It is auto incremented so whenever a new game is added, id is automatically updated.

Title refers to the name of the game, VARCHAR data type is used, and it is UNIQUE and NOT NULL for the usual reasons.

Price$ refers to the price of the game DECIMAL data type is used as it is better than INT for prices. It is NOT NULL and the default is set to 0.00 as if the price is 0.00, it means the game is free.

Type is foreign key, it has INT data type, is NOT NULL and references to GenreID of Genres. Each game belongs to a genre and genres have games.

Publisher is foreign key, it has INT data type, is NOT NULL and references to PublisherID of Publishers. Each game comes from a publisher and publishers have many games.

3.4.2 Screenshot of table creation

```
MariaDB [OnlineGameDistributionService]> CREATE TABLE Games(GameID INT PRIMARY KEY AUTO_INCREMENT, Title VARCHAR(255)
 NOT NULL UNIQUE, Price$ DECIMAL(10,2) NOT NULL DEFAULT'0.00', Type INT NOT NULL, Publisher INT NOT NULL, FOREIGN KEY
 (Type) REFERENCES Genres(GenreID), FOREIGN KEY (Publisher) REFERENCES Publishers(PublisherID));
Query OK, 0 rows affected (0.264 sec)
```

Figure 15: Creating Games table

3.4.3 Description of table

```
MariaDB [OnlineGameDistributionService]> DESCRIBE Games;
+-----------+---------------+------+-----+---------+----------------+
| Field     | Type          | Null | Key | Default | Extra          |
+-----------+---------------+------+-----+---------+----------------+
| GameID    | int(11)       | NO   | PRI | NULL    | auto_increment |
| Title     | varchar(255)  | NO   | UNI | NULL    |                |
| Price$    | decimal(10,2) | NO   |     | 0.00    |                |
| Type      | int(11)       | NO   | MUL | NULL    |                |
| Publisher | int(11)       | NO   | MUL | NULL    |                |
+-----------+---------------+------+-----+---------+----------------+
5 rows in set (0.017 sec)
```

Figure 16: Description of games

3.4.4 Screenshot of data insertion

```
MariaDB [OnlineGameDistributionService]> INSERT INTO Games VALUES(100, "DOTA 2", "", 34, 1902),
    -> ("", "Hearthstone", "", 49, 3648),
    -> ("", "Minecraft", 26.95, 7, 7777),
    -> ("", "Overwatch", 19.99, 3, 3648),
    -> ("", "Witcher 3", 49.99, 20, 8969),
    -> ("", "CS:GO","", 3, 1902);
Query OK, 6 rows affected, 8 warnings (0.795 sec)
Records: 6  Duplicates: 0  Warnings: 8
```

Figure 17: Inserting data into Games

3.4.5 Selecting all inserted data

```
MariaDB [OnlineGameDistributionService]> SELECT*FROM GAMES;
+--------+-------------+--------+------+-----------+
| GameID | Title       | Price$ | Type | Publisher |
+--------+-------------+--------+------+-----------+
|    100 | DOTA 2      |   0.00 |   34 |      1902 |
|    101 | Hearthstone |   0.00 |   49 |      3648 |
|    102 | Minecraft   |  26.95 |    7 |      7777 |
|    103 | Overwatch   |  19.99 |    3 |      3648 |
|    104 | Witcher 3   |  49.99 |   20 |      8969 |
|    105 | CS:GO       |   0.00 |    3 |      1902 |
+--------+-------------+--------+------+-----------+
6 rows in set (0.000 sec)
```

Figure 18: Selecting all data from games

3.5 Purchases

3.5.1 Explanation

This table contains the information about the purchases. Following are the attributes:

- PurchaseID
- PurchaseDate
- User

PurchaseID is the primary key, it uniquely identifies each purchase and has INT datatype and is auto incremented for ease of use.

PurchaseDate is the date of purchase, DATE is the suitable data type and it cannot be null.

User has INT value, it is a foreign key and references to the UserID of users table. It cannot be null because it is referencing to a user so making it null would defeat the purpose. A single user can have many purchases.

3.5.2 Screenshot of table creation

```
MariaDB [OnlineGameDistributionService]> CREATE TABLE Purchases(PurchaseID INT PRIMARY KEY AUTO_INCREMENT, PurchaseDate DATE NOT NULL, User INT NOT
NULL, FOREIGN KEY(user) REFERENCES Users (UserID));
Query OK, 0 rows affected (0.284 sec)
```

Figure 19: Creating Purchases table

3.5.3 Description of table

```
MariaDB [OnlineGameDistributionService]> DESCRIBE Purchases;
+--------------+---------+------+-----+---------+----------------+
| Field        | Type    | Null | Key | Default | Extra          |
+--------------+---------+------+-----+---------+----------------+
| PurchaseID   | int(11) | NO   | PRI | NULL    | auto_increment |
| PurchaseDate | date    | NO   |     | NULL    |                |
| User         | int(11) | NO   | MUL | NULL    |                |
+--------------+---------+------+-----+---------+----------------+
3 rows in set (0.019 sec)
```

Figure 20: Description of Purchases

3.5.4 Screenshot of data insertion

```
MariaDB [OnlineGameDistributionService]> INSERT INTO Purchases VALUES(194001,"2019-12-9",1000),
    -> ("","2019-12-9",1001),
    -> ("","2019-12-10",1002),
    -> ("","2019-12-10",1002),
    -> ("","2019-12-10",1003);
Query OK, 5 rows affected, 4 warnings (0.136 sec)
Records: 5  Duplicates: 0  Warnings: 4
```

Figure 21: Inserting data into Purchases

3.5.5 Selecting all inserted data

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Purchases;
+------------+--------------+------+
| PurchaseID | PurchaseDate | User |
+------------+--------------+------+
|     194001 | 2019-12-09   | 1000 |
|     194002 | 2019-12-09   | 1001 |
|     194003 | 2019-12-10   | 1002 |
|     194004 | 2019-12-10   | 1002 |
|     194005 | 2019-12-10   | 1003 |
+------------+--------------+------+
5 rows in set (0.000 sec)
```

Figure 22: Selecting all data from Purchases

3.6 OrderedGames

3.6.1 Explanation

This table contains information about the orders i.e. which game belongs to which purchase. Multiple games can belong to a purchase. Following are the attributes:

- OrderID
- Purchase
- Game

OrderID is the primary key and has INT data type. It is auto incremented for ease of use. Each order can be tracked with its help.

Purchase has INT data type, it is a foreign key and references to PurchaseID of Purchases table. Each ordered game belongs to a purchase that is why this attribute is needed. This attribute is made NOT NULL.

Game also has INT data type and is a foreign key. It references to GameID of games table because each order needs a game. This attribute is made NOT NULL.

3.6.2 Screenshot of table creation

```
MariaDB [OnlineGameDistributionService]> CREATE TABLE OrderedGames(OrderID INT PRIMARY KEY AUTO_INCREMENT, Purchase INT NOT NULL, Game INT NOT NULL,
FOREIGN KEY(Purchase) REFERENCES Purchases (PurchaseID), FOREIGN KEY(Game) REFERENCES Games (gameID));
Query OK, 0 rows affected (0.693 sec)
```

Figure 23: Creating OrderedGames table

3.6.3 Description of table

```
MariaDB [OnlineGameDistributionService]> DESCRIBE OrderedGames;
+----------+---------+------+-----+---------+----------------+
| Field    | Type    | Null | Key | Default | Extra          |
+----------+---------+------+-----+---------+----------------+
| OrderID  | int(11) | NO   | PRI | NULL    | auto_increment |
| Purchase | int(11) | NO   | MUL | NULL    |                |
| Game     | int(11) | NO   | MUL | NULL    |                |
+----------+---------+------+-----+---------+----------------+
3 rows in set (0.141 sec)
```

Figure 24: Description of OrderedGames

3.6.4 Screenshot of data insertion

```
MariaDB [OnlineGameDistributionService]> INSERT INTO OrderedGames Values(1,194001,100)
    -> ,("",194002,101),
    -> ("",194003,102),
    -> ("",194004,103),
    -> ("",194005,103);
Query OK, 5 rows affected, 4 warnings (0.141 sec)
Records: 5  Duplicates: 0  Warnings: 4
```

Figure 25: Inserting data into OrderedGames

3.6.5 Selecting all inserted data

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM OrderedGames;
+---------+----------+------+
| OrderID | Purchase | Game |
+---------+----------+------+
|       1 |   194001 |  100 |
|       2 |   194002 |  101 |
|       3 |   194003 |  102 |
|       4 |   194004 |  103 |
|       5 |   194005 |  103 |
+---------+----------+------+
5 rows in set (0.000 sec)
```

Figure 26: Selecting all data from OrderedGames

4 Data Dictionary

4.1 Data Dictionary for Users

Entity name	Entity description	Column Name	Column Description	Data type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Users	Someone who uses the service	UserID	ID to separate users	INT	11	True	False	False	True	Auto Incremented
		Name	Name of the users	VARCHAR	50	False	False	False	False	
		Email	Email to contact the users	VARCHAR	255	False	False	False	True	

4.2 Data Dictionary for Genres

Entity name	Entity description	Column Name	Column Description	Data type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Genres	The type of game	GenreID	ID to separate genres	INT	11	True	False	False	True	Auto Incremented
		Genre	Name of the genres	VARCHAR	255	False	False	False	True	

4.3 Data Dictionary for Publishers

Entity name	Entity description	Column Name	Column Description	Data type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Publishers	Company to publish the game	PublisherID	ID for publisher	INT	11	True	False	False	True	
		Name	Name of publisher	VARCHAR	50	False	False	False	True	
		Website	Website of publisher	VARCHAR	255	False	False	False	True	

4.4 Data Dictionary for Games

Entity name	Entity description	Column Name	Column Description	Data type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Games	Purchasable games	Game ID	Unique ID for each game	INT	11	True	False	False	True	Auto Incremented
		Title	Name of the game	VARCHAR	255	False	False	False	True	
		Price$	Price of the game	DOUBLE	10,2	False	False	False	False	Default=0.00
		Type	Genre of the game	INT	11	False	True	False	False	References to GenreID of Genres table
		Publisher	Who published the game	INT	11	False	True	False	False	References to PublisherID of Publishers table

4.5 Data Dictionary for Purchases

Entity name	Entity description	Column Name	Column Description	Data type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Purchases	Purchases made by the users	PurchaseID	ID for purchase	INT	11	True	False	False	True	Auto incremented
		PurchaseDate	Date of purchase	DATE		False	False	False	True	
		User	User who made the purchase	INT	11	False	True	False	True	References to UserID of Users

4.6 Data Dictionary for OrderedGames

Entity name	Entity description	Column Name	Column Description	Data type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
OrderedGames	Games that are about to be bought	OrderID	ID for ordered games	INT	11	True	False	False	True	Auto incremented
		Game	Game to be bought	INT	11	False	True	False	False	References to GameID of Games table
		Purchase	Belongs to which purchase	INT	11	False	True	False	False	References to PurchaseID of Purchases table

5 Queries

Example of different queries

(Note that there can be multiple ways to use the queries with different syntax)

5.1 BETWEEN

When we need to select values in a given range, BETWEEN is used. The syntax is:

- SELECT * FROM *table_name* Where *column_name* BETWEEN value1 AND value2;

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM games WHERE Price$ BETWEEN 0 AND 20;
+--------+-------------+--------+------+-----------+
| GameID | Title       | Price$ | Type | Publisher |
+--------+-------------+--------+------+-----------+
|    100 | DOTA 2      |   0.00 |   34 |      1902 |
|    101 | Hearthstone |   0.00 |   49 |      3648 |
|    103 | Overwatch   |  19.99 |    3 |      3648 |
|    105 | CS:GO       |   0.00 |    3 |      1902 |
+--------+-------------+--------+------+-----------+
4 rows in set (0.000 sec)
```

Figure 27: BETWEEN

In the above example, BETWEEN is used to show the data of games whose price ranges from 0 to 20.

5.2 ORDER BY

When we need to sort the results in ascending or descending order, ORDER BY is used. By default, it is in ascending order. The syntax is:

- SELECT * FROM *table_name* ORDER BY *column_name* AESC | DESC;

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Users ORDER BY Name DESC;
+--------+---------+------------------+
| UserID | Name    | Email            |
+--------+---------+------------------+
|   1002 | Saitama | saitama@mail.com |
|   1000 | Reigen  | reigen@mail.com  |
|   1001 | Mob     | mob@mail.com     |
|   1003 | Genos   | genos@mail.com   |
|   1004 | Dimple  | dimple@mail.com  |
+--------+---------+------------------+
5 rows in set (0.000 sec)
```

Figure 28: ORDER BY

In the above example; ORDER BY arranges the data in descending order according to name.

5.3 IN

IN is used to specify multiple values in WHERE clause. IN omits the use of OR. The syntax is:

- SELECT * FROM *table_name* WHERE *column_name* IN (value1, value2 …);

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Purchases WHERE User IN(1000,1002);
+------------+--------------+------+
| PurchaseID | PurchaseDate | User |
+------------+--------------+------+
|     194001 | 2019-12-09   | 1000 |
|     194003 | 2019-12-10   | 1002 |
|     194004 | 2019-12-10   | 1002 |
+------------+--------------+------+
3 rows in set (0.104 sec)
```

Figure 29: IN

Here, IN is used to check the purchases of user 1000 and 1002.

5.4 LIKE

LIKE is used in WHERE clause to search for patterns. The syntax is:

- SELECT * FROM *table_name* WHERE *column_name* LIKE “value%” | “%value% | “%value”;

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Genres WHERE Genre Like "S%";
+---------+-----------------------+
| GenreID | Genre                 |
+---------+-----------------------+
|      24 | SandboxRPG            |
|       3 | Shooter               |
|      26 | Simulation            |
|      42 | Sports                |
|      47 | Sports-based fighting |
|       6 | Stealth               |
|      30 | Strategy              |
|       7 | Survival              |
|      11 | Survival Horror       |
+---------+-----------------------+
9 rows in set (0.001 sec)
```

Figure 30: Like

Here, LIKE is used to select all the data from Genres where Genre begins with S.

5.5 LIMIT

LIMIT is used to determine how many data to show as a result of using query. The syntax is:

- ----------*QUERY*---------- LIMIT value;

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM OrderedGames ORDER BY Purchase DESC LIMIT 2;
+---------+----------+------+
| OrderID | Purchase | Game |
+---------+----------+------+
|       5 |   194005 |  103 |
|       4 |   194004 |  103 |
+---------+----------+------+
2 rows in set (0.000 sec)
```

Figure 31: LIMIT

In the above case, LIMIT is used to only show two records after using a query to list all the data in descending order according to purchase.

5.6 COUNT

COUNT is used to find total number of records in a table. COUNT(*) returns the total number of rows in a table COUNT(*column_name*) returns total number of data excluding null. The syntax is:

- SELECT COUNT(* | *column_name*) FROM *table_name*;

```
MariaDB [OnlineGameDistributionService]> SELECT COUNT(Genre) AS Total_Genres FROM Genres;
+--------------+
| Total_Genres |
+--------------+
|           50 |
+--------------+
1 row in set (0.001 sec)
```

Figure 32: COUNT

In the above case; COUNT is used to count the number of data in Genre of Genres table AS is used to give a temporary name.

5.7 GROUP BY

GROUP BY groups the rows according to same values in a column GROUP BY is used with aggregate functions like (COUNT, MAX, SUM etc.). The syntax is:

- *QUERY COMBINATION* + GROUP BY

```
MariaDB [OnlineGameDistributionService]> SELECT*FROM GAMES;
+--------+-------------+--------+------+-----------+
| GameID | Title       | Price$ | Type | Publisher |
+--------+-------------+--------+------+-----------+
|    100 | DOTA 2      |   0.00 |   34 |      1902 |
|    101 | Hearthstone |   0.00 |   49 |      3648 |
|    102 | Minecraft   |  26.95 |    7 |      7777 |
|    103 | Overwatch   |  19.99 |    3 |      3648 |
|    104 | Witcher 3   |  49.99 |   20 |      8969 |
|    105 | CS:GO       |   0.00 |    3 |      1902 |
+--------+-------------+--------+------+-----------+
6 rows in set (0.000 sec)
```

Figure 33: Data from Games

```
MariaDB [OnlineGameDistributionService]> SELECT Publisher, COUNT(*) AS Total_Games FROM Games Group BY Publisher;
+-----------+-------------+
| Publisher | Total_Games |
+-----------+-------------+
|      1902 |           2 |
|      3648 |           2 |
|      7777 |           1 |
|      8969 |           1 |
+-----------+-------------+
4 rows in set (0.000 sec)
```

Figure 34: GROUP BY

In the above example; we use group by along with count to show the number of games from each publishers. We count how many times a publisher is repeated and group the data according to unique publishers to get the number of games form each publisher.

5.8 HAVING

HAVING is like WHERE, HAVING is use to filter data after aggregate function. The syntax is:

- *QUERY* + HAVING

```
MariaDB [OnlineGameDistributionService]> SELECT*FROM GAMES;
+--------+-------------+--------+------+-----------+
| GameID | Title       | Price$ | Type | Publisher |
+--------+-------------+--------+------+-----------+
|    100 | DOTA 2      |   0.00 |   34 |      1902 |
|    101 | Hearthstone |   0.00 |   49 |      3648 |
|    102 | Minecraft   |  26.95 |    7 |      7777 |
|    103 | Overwatch   |  19.99 |    3 |      3648 |
|    104 | Witcher 3   |  49.99 |   20 |      8969 |
|    105 | CS:GO       |   0.00 |    3 |      1902 |
+--------+-------------+--------+------+-----------+
6 rows in set (0.000 sec)
```

Figure 35: Data from Games

```
MariaDB [OnlineGameDistributionService]> SELECT Publisher, COUNT(*) AS Total_Games FROM Games Group BY Publisher;
+-----------+-------------+
| Publisher | Total_Games |
+-----------+-------------+
|      1902 |           2 |
|      3648 |           2 |
|      7777 |           1 |
|      8969 |           1 |
+-----------+-------------+
4 rows in set (0.000 sec)
```

Figure 36: Grouped data

```
MariaDB [OnlineGameDistributionService]> SELECT Publisher, COUNT(*) AS Total_Games FROM Games Group BY Publisher HAVING Total_Games>1;
+-----------+-------------+
| Publisher | Total_Games |
+-----------+-------------+
|      1902 |           2 |
|      3648 |           2 |
+-----------+-------------+
2 rows in set (0.000 sec)
```

Figure 37: HAVING

In the above example, data from a table is sorted out and grouped and again HAVING is used to give a condition before finally displaying the data after using an aggregate function.

5.9 DISTINCT

DISTINCT gives inly the unique values in a column. The syntax is:

- SELECT DISTINCT *column_name* FROM *table_name*;

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Purchases;
+------------+--------------+------+
| PurchaseID | PurchaseDate | User |
+------------+--------------+------+
|     194001 | 2019-12-09   | 1000 |
|     194002 | 2019-12-09   | 1001 |
|     194003 | 2019-12-10   | 1002 |
|     194004 | 2019-12-10   | 1002 |
|     194005 | 2019-12-10   | 1003 |
+------------+--------------+------+
5 rows in set (0.000 sec)
```

Figure 38: Data from Purchases

```
MariaDB [OnlineGameDistributionService]> SELECT DISTINCT User FROM Purchases;
+------+
| User |
+------+
| 1000 |
| 1001 |
| 1002 |
| 1003 |
+------+
4 rows in set (0.000 sec)
```

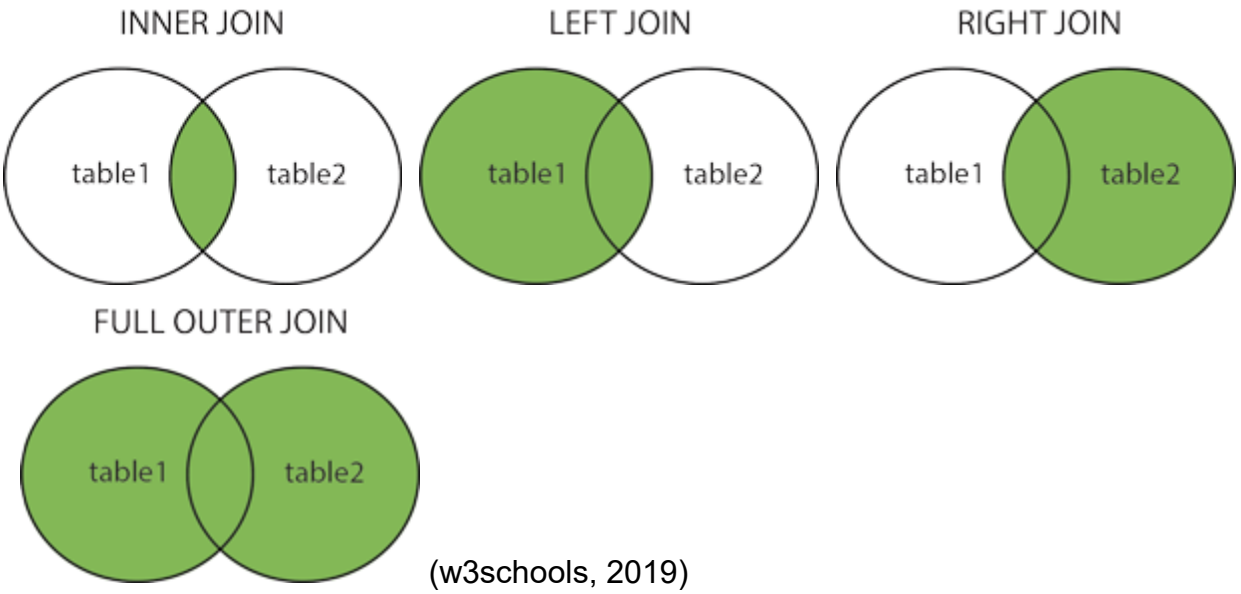
Figure 39: DISTINCT

Here, DISTINCT is used to select unique users from the purchases table.

5.10 JOIN

Here are the different types of the JOINs in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



(w3schools, 2019)

5.10.1 INNER JOIN

Syntax:

- SELECT column_*name* FROM *table1* INNER JOIN *table2* ON *table1.column_name=table2.column_name;*

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Publishers JOIN Games ON Publishers.PublisherID=Games.Publisher;
+-------------+------------------------+-------------------+--------+-------------+--------+------+-----------+
| PublisherID | Name                   | Website           | GameID | Title       | Price$ | Type | Publisher |
+-------------+------------------------+-------------------+--------+-------------+--------+------+-----------+
|        1902 | Valve                  | valvesoftware.com |    100 | DOTA 2      |   0.00 |   34 |      1902 |
|        3648 | Blizzard Entertainment | blizzard.com      |    101 | Hearthstone |   0.00 |   49 |      3648 |
|        7777 | Mojang                 | mojang.com        |    102 | Minecraft   |  26.95 |    7 |      7777 |
|        3648 | Blizzard Entertainment | blizzard.com      |    103 | Overwatch   |  19.99 |    3 |      3648 |
|        8969 | CD Projekt             | cdprojekt.com     |    104 | Witcher 3   |  49.99 |   20 |      8969 |
|        1902 | Valve                  | valvesoftware.com |    105 | CS:GO       |   0.00 |    3 |      1902 |
+-------------+------------------------+-------------------+--------+-------------+--------+------+-----------+
6 rows in set (0.001 sec)
```

Figure 40: Joining all data from Games and Publishers

```
MariaDB [OnlineGameDistributionService]> SELECT Games.Title,Publishers.Name AS Name_Of_Pblisher FROM Publishers INNER JOIN Games ON Publishers.Publi
sherID=Games.Publisher;
+-------------+------------------------+
| Title       | Name_Of_Pblisher       |
+-------------+------------------------+
| DOTA 2      | Valve                  |
| Hearthstone | Blizzard Entertainment |
| Minecraft   | Mojang                 |
| Overwatch   | Blizzard Entertainment |
| Witcher 3   | CD Projekt             |
| CS:GO       | Valve                  |
+-------------+------------------------+
6 rows in set (0.001 sec)
```

Figure 41: INNER JOIN

Here, INNER JOIN is used to only show title of game and name of publisher when Publisher from games matches to PublisherID.

5.10.2 LEFT JOIN

Syntax:

- SELECT *column_name* FROM *table1* LEFT JOIN *table2* ON *table1.column_name=table2.column_name;*

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Publishers;
+-------------+------------------------+--------------------+
| PublisherID | Name                   | Website            |
+-------------+------------------------+--------------------+
|        1222 | Riot Games             | riotgames.com      |
|        1902 | Valve                  | valvesoftware.com  |
|        3648 | Blizzard Entertainment | blizzard.com       |
|        7777 | Mojang                 | mojang.com         |
|        8969 | CD Projekt             | cdprojekt.com      |
+-------------+------------------------+--------------------+
5 rows in set (0.116 sec)
```

Figure 42: Data from Publishers

```
MariaDB [OnlineGameDistributionService]> SELECT*FROM GAMES;
+--------+-------------+--------+------+-----------+
| GameID | Title       | Price$ | Type | Publisher |
+--------+-------------+--------+------+-----------+
|    100 | DOTA 2      |   0.00 |   34 |      1902 |
|    101 | Hearthstone |   0.00 |   49 |      3648 |
|    102 | Minecraft   |  26.95 |    7 |      7777 |
|    103 | Overwatch   |  19.99 |    3 |      3648 |
|    104 | Witcher 3   |  49.99 |   20 |      8969 |
|    105 | CS:GO       |   0.00 |    3 |      1902 |
+--------+-------------+--------+------+-----------+
6 rows in set (0.000 sec)
```

Figure 43: Data from Games

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Publishers LEFT JOIN Games ON Publishers.PublisherID=Games.Publisher;
+-------------+-----------------------+-------------------+--------+-------------+--------+------+-----------+
| PublisherID | Name                  | Website           | GameID | Title       | Price$ | Type | Publisher |
+-------------+-----------------------+-------------------+--------+-------------+--------+------+-----------+
|        1222 | Riot Games            | riotgames.com     |   NULL | NULL        |   NULL | NULL |      NULL |
|        1902 | Valve                 | valvesoftware.com |    100 | DOTA 2      |   0.00 |   34 |      1902 |
|        1902 | Valve                 | valvesoftware.com |    105 | CS:GO       |   0.00 |    3 |      1902 |
|        3648 | Blizzard Entertainment | blizzard.com      |    101 | Hearthstone |   0.00 |   49 |      3648 |
|        3648 | Blizzard Entertainment | blizzard.com      |    103 | Overwatch   |  19.99 |    3 |      3648 |
|        7777 | Mojang                | mojang.com        |    102 | Minecraft   |  26.95 |    7 |      7777 |
|        8969 | CD Projekt            | cdprojekt.com     |    104 | Witcher 3   |  49.99 |   20 |      8969 |
+-------------+-----------------------+-------------------+--------+-------------+--------+------+-----------+
7 rows in set (0.000 sec)
```

Figure 44: LEFT JOIN

LEFT JOIN shows all the data of Publishers and matching data from Games NULL means no data matches.

5.10.3 RIGHT JOIN

Syntax:

- SELECT *column_name* FROM *table1* RIGHT JOIN *table2* ON *table1.column_name=table2.column_name;*

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Publishers;
+-------------+-------------------------+-------------------+
| PublisherID | Name                    | Website           |
+-------------+-------------------------+-------------------+
|        1222 | Riot Games              | riotgames.com     |
|        1902 | Valve                   | valvesoftware.com |
|        3648 | Blizzard Entertainment | blizzard.com      |
|        7777 | Mojang                  | mojang.com        |
|        8969 | CD Projekt              | cdprojekt.com     |
+-------------+-------------------------+-------------------+
5 rows in set (0.116 sec)
```

Figure 45: Data from Publishers

```
MariaDB [OnlineGameDistributionService]> SELECT*FROM GAMES;
+--------+-------------+--------+------+-----------+
| GameID | Title       | Price$ | Type | Publisher |
+--------+-------------+--------+------+-----------+
|    100 | DOTA 2      |   0.00 |   34 |      1902 |
|    101 | Hearthstone |   0.00 |   49 |      3648 |
|    102 | Minecraft   |  26.95 |    7 |      7777 |
|    103 | Overwatch   |  19.99 |    3 |      3648 |
|    104 | Witcher 3   |  49.99 |   20 |      8969 |
|    105 | CS:GO       |   0.00 |    3 |      1902 |
+--------+-------------+--------+------+-----------+
6 rows in set (0.000 sec)
```

Figure 46: Data from Games

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Publishers RIGHT JOIN Games ON Publishers.PublisherID=Games.Publisher;
+-------------+------------------------+------------------+--------+-------------+--------+------+-----------+
| PublisherID | Name                   | Website          | GameID | Title       | Price$ | Type | Publisher |
+-------------+------------------------+------------------+--------+-------------+--------+------+-----------+
|        1902 | Valve                  | valvesoftware.com |    100 | DOTA 2      |   0.00 |   34 |      1902 |
|        3648 | Blizzard Entertainment | blizzard.com      |    101 | Hearthstone |   0.00 |   49 |      3648 |
|        7777 | Mojang                 | mojang.com        |    102 | Minecraft   |  26.95 |    7 |      7777 |
|        3648 | Blizzard Entertainment | blizzard.com      |    103 | Overwatch   |  19.99 |    3 |      3648 |
|        8969 | CD Projekt             | cdprojekt.com     |    104 | Witcher 3   |  49.99 |   20 |      8969 |
|        1902 | Valve                  | valvesoftware.com |    105 | CS:GO       |   0.00 |    3 |      1902 |
+-------------+------------------------+------------------+--------+-------------+--------+------+-----------+
6 rows in set (0.001 sec)
```

Figure 47: RIGHT JOIN

Here all the games have publishers so Publisher and PublisherID always match and all data is shown from both tables.

5.10.4 OUTER JOIN

OUTER JOIN can be implemented by combining the result of LEFT JOIN and RIGHT JOIN. Syntax:

- SELECT *column_name* FROM *table1* LEFT JOIN *table2* ON *table1.column_name=table2.column_name;*

UNION

SELECT *column_name* FROM *table1* RIGHT JOIN *table2* ON *table1.column_name=table2.column_name;*

```
MariaDB [OnlineGameDistributionService]> SELECT * FROM Publishers LEFT JOIN Games ON Publishers.PublisherID=Games.Publisher
    -> UNION
    -> SELECT * FROM Publishers RIGHT JOIN Games ON Publishers.PublisherID=Games.Publisher;
+-------------+-----------------------+------------------+--------+-------------+--------+------+-----------+
| PublisherID | Name                  | Website          | GameID | Title       | Price$ | Type | Publisher |
+-------------+-----------------------+------------------+--------+-------------+--------+------+-----------+
|        1222 | Riot Games            | riotgames.com    |   NULL | NULL        |   NULL | NULL |      NULL |
|        1902 | Valve                 | valvesoftware.com |    100 | DOTA 2      |   0.00 |   34 |      1902 |
|        1902 | Valve                 | valvesoftware.com |    105 | CS:GO       |   0.00 |    3 |      1902 |
|        3648 | Blizzard Entertainment | blizzard.com     |    101 | Hearthstone |   0.00 |   49 |      3648 |
|        3648 | Blizzard Entertainment | blizzard.com     |    103 | Overwatch   |  19.99 |    3 |      3648 |
|        7777 | Mojang                | mojang.com       |    102 | Minecraft   |  26.95 |    7 |      7777 |
|        8969 | CD Projekt            | cdprojekt.com    |    104 | Witcher 3   |  49.99 |   20 |      8969 |
+-------------+-----------------------+------------------+--------+-------------+--------+------+-----------+
7 rows in set (0.003 sec)
```

Figure 48: OUTER JOIN

The above figure shows the result of outer join.

6 CONCLUSION

6.1 Research and Conclusion

Databases are inseparable part of our daily lives, everything needs a database to function. This project showed a simple database for a game distribution service and highlighted the use of different queries to sort the data that we may need. Through the project I also personally learned a lot of things about how databases and database management software work and how the same query can be used with another syntax or another query. Research was done mostly through internet and books and the database was thought of as a simple idea for the project. In conclusion the project has been a success in showing the creation and usage of a database.

Bibliography

©Cambridge University Press. (2019) *Cambridge Dictionary* [Online]. Available from: https://dictionary.cambridge.org/dictionary/english/data [Accessed 17 November 2019].

ENCYCLOPEDIA Britannica©. (2013) *Britannica REFERENCE ENCYCLOPEDIA*. Singapore: ENCYCLOPEDIA Britannica©.

Foote, K.D. (2017) *Dataversity* [Online]. Available from: https://www.dataversity.net/brief-history-database-management/# [Accessed 12 December 2019].

GeeksforGeeks. (2019) *GeeksforGeeks* [Online]. Available from: https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/ [Accessed 17 November 2019].

w3schools. (2019) *SQL Joins* [Online]. Available from: https://www.w3schools.com/sql/sql_join.asp [Accessed 18 December 2019].