1 Find and critique a dataset

1.1 Access the Dataset

Mobile games industry is worth billions of dollars, with company spending lots of money on the development and marketing of these games to equally large market. Games are planned experience for players where some provide significant intellectual challenge and the opportunity to plan and carry out a myriad of strategies. Other games provide players with emotional experiences like fun, tension and even fear.

Strategy game does not rely on luck but strategy in determining the course of the game. This means that player need to use decision making skills to have a high significance in determining the outcome of the games. The main elements of the strategy game are highly thematic with gameplay that emphasizes player decision-making with multiple paths to victory and multiple choices per turn. Successful games match players' expectations to the choices made by the designer. Using this dataset, insights can be gained into a sub-market of this market, strategy games.

1.2 Choice of dataset

The dataset I have chosen the data of 17007 strategy games on the Apple App Store. It was collected on the 3rd of August 2019, using the iTunes API and the App Store sitemap. The data contain the details of the application such as size, price, and developer etc.

1.3 Quality

This dataset is uploaded by Tristan and published on Kaggle. The data was collected with iTunes API and the App Store sitemap. Thus, the reliability of the dataset is highly reliable as it is collected by the Apple App Store tool.

1.4 Details

The dataset contains a wide range of details of the application which might affect the rating of the application. This is helpful in finding the overall success pf a game and take into consideration for future making. The data in the dataset were too details hence there is a few columns must be removing such as URL and icon URL as it will be difficult for us to include the data.

1.5 Documentation

The dataset is collected from Apple Store official site. Despite the reliability and the explanation of every attribute given by the publisher, to understand the data, we need to have the basics of musical knowledge. Otherwise, we might not understand the meaning of the value.

1.6 Interrelation

This dataset can be used to compare the higher rating apps on other platforms such as google store. This can help to analyse which platform have higher demand for that app. Not only so, but it can also use to compare with other type of games in Apple Store. This allows us to the demand of the strategy game app and what are the factors affecting the user interest of an application.

1.7 Use

This dataset can be used to analyse what are the factors contribute to the popularity of the game application. However, it does not provide more details such as uninstallation rates. Hence, we cannot tell if the user still continues to use the application.

1.8 Discoverability

There are very less datasets like this available on internet. Many of them do not provide details information of the application.

1.9 Interests of the dataset

A question I would like to ask is “What are the most popular genres in the strategy games?”. Due to the competitiveness of the game design industry, it is important for game designer to not only think out of the box so it will be unique and increase the attention from user. However, it is also important to keep in mind what are the trend among the users so understand the most popular genres keep the game designer in trend. Another question I would like to ask is “Will users be more interested in newly launched game application whenever there is one?”. This brings back to the competitiveness of the game design industry, users might not stay with the application for long has there is always newly launched game which might more interesting and attractive. The last question will be “Is free mobile game application always more popular than paid app”. This can help the game designer to think if they should make their future app free.

The questions are as follow:

> **Question 1**: “What are the most popular genres in the strategy games?”
>
> **Question 2:** “Will users be more interested in newly launched game application whenever there is one?”
>
> **Question 3:** “Is free mobile game application always more popular than paid application?”

2 Model your data

2.1 The dataset

Simple data processing was done to remove unnecessary column, nan values and duplicate rows. This was done with python.

Table below shows all the attributes present in the original dataset. However, we are only using a subset of the data on attributes that are possible factors of mobile strategy rating. Based on that the data is further filtered to only use attributes with filled data, meaning attributes with many empty values are dropped, meaningless column such as URL and repeated attributes such as app id are not used as well.

Attributes	Descriptions
URL	App URL
ID	Extracted from App URL
Name	App name
Subtitle	App subtitle
Icon URL	App icon URL
In-app Purchases	Price in app purchase
Description	App description
Average User rating	App rating rounded to nearest .5, requires at least 5 ratings
User Rating Count	Number of ratings internationally
Price	Price in USD
Developer	The developer of the app
Age Rating	App age rating in Either 4+, 9+, 12+ or 17+
Languages	The available language for the app
Size	Size of the app in bytes
Primary genre	The main genre
Genres	The sub-genre
Original Release Date	When the app released
Current version Released Date	When was the app last updated

2.2 Normalisation

The table were normalised to 1NF form by ensuring the column with more than one data are splited with python code. The first data frame is before normalise, the one below is after normalise.

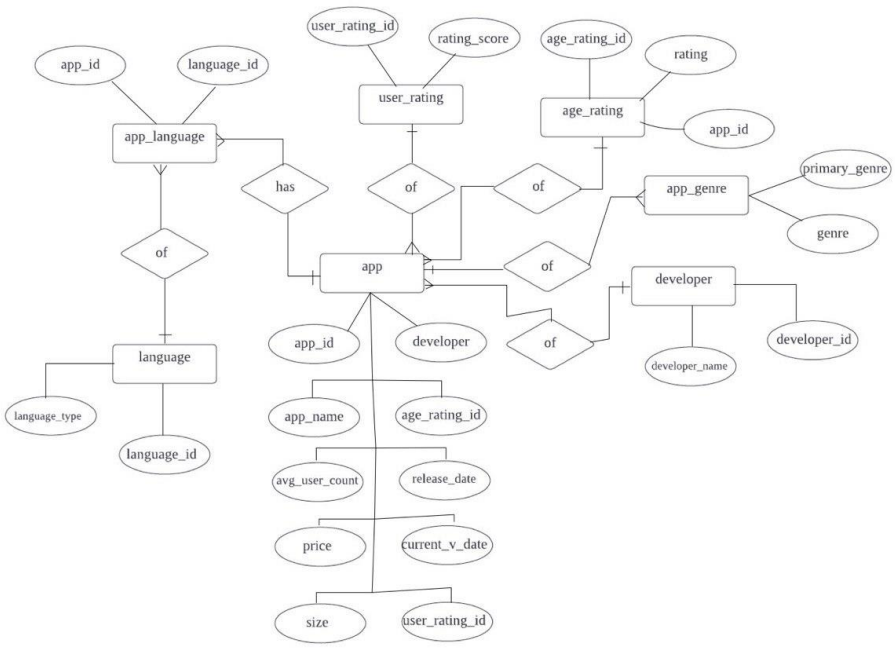
	Name	Average User Rating	User Rating Count	Price	Developer	Age Rating	Languages	Size	Primary Genre	Genres	Original Release Date	Current Version Release Date
0	Sudoku	4.0	3553.0	2.99	Mighty Mighty Good Games	4+	DA, NL, EN, FI, FR, DE, IT, JA, KO, NB, PL, PT...	15853568.0	Games	Games, Strategy, Puzzle	11/07/2008	30/05/2017
1	Reversi	3.5	284.0	1.99	Kiss The Machine	4+	EN	12328960.0	Games	Games, Strategy, Board	11/07/2008	17/05/2018

	Name	Average User Rating	User Rating Count	Price	Developer	Age Rating	Languages	Size	Primary Genre	Genres	Original Release Date	Curren Versio Releas Dat
13	Marple	3.5	989	0.99	Mikko Kankainen	4+	EN	3643392.0	Games	Games	2008-08-28	2019 05-0
13	Marple	3.5	989	0.99	Mikko Kankainen	4+	EN	3643392.0	Games	Puzzle	2008-08-28	2019 05-0
13	Marple	3.5	989	0.99	Mikko Kankainen	4+	EN	3643392.0	Games	Strategy	2008-08-28	2019 05-0

The tables are then in normalised in fourth normalised form (4NF)to form a fact table. However, when performing analytics tasks, we would require the tables to be joined together (demoralised) to create a suitable table to perform predictive analytics.

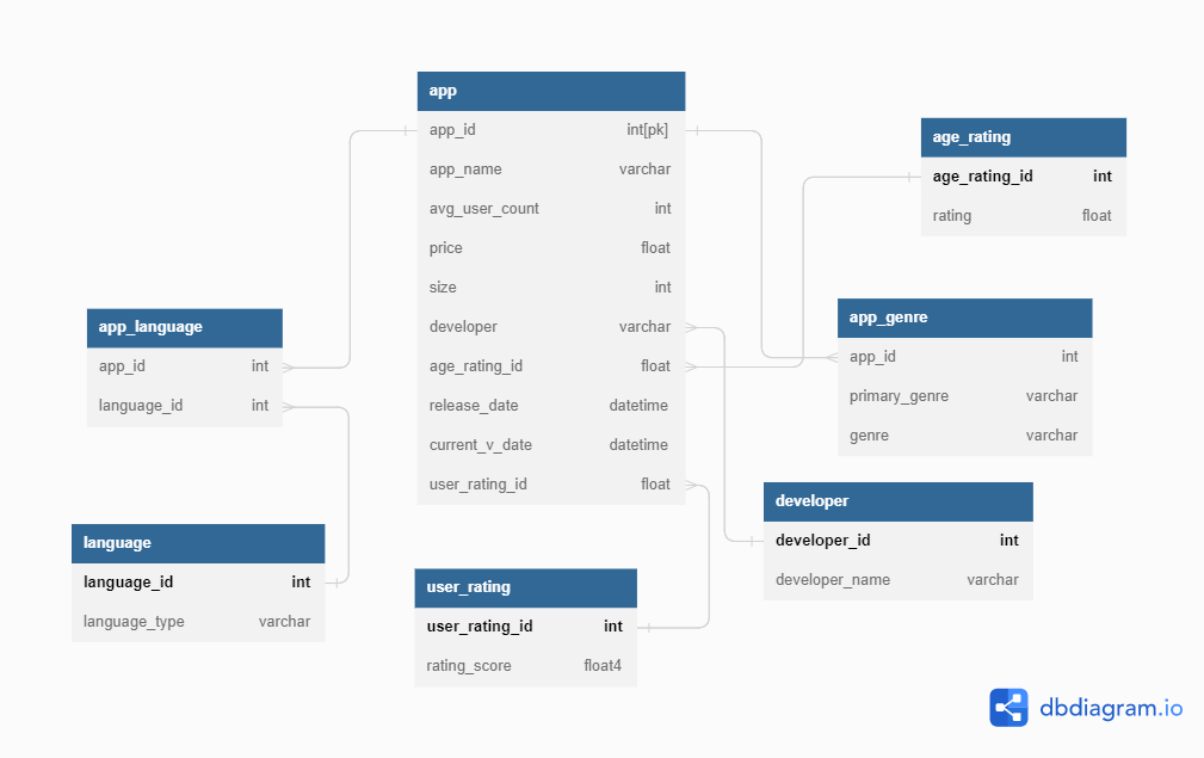
2.3 E/R model

Below shows the ER model of the data where the table of app have a one to many relationship with the other tables, linked by foreign keys. For example, many entries for app can have the diff app language which is linked by the app_id.



2.4 E/R Diagram with cardinality

Figure further represents how the tables are connected to each other with the use of foreign keys.



3 Create the database

3.1 Record all CREATE commands

All create commands are made in the mySQL console in the terminal. Firstly, a database called mobileApp is created:

```
CREATE DATABASE mobileApp;
```

Secondly, we create a database user called ‘user’ to grant access to the database mobileApp we created previously:

```
CREATE USER ‘kelly@’%’ IDENTIFIED WITH mysql_native_password BY ‘sqlpw
```

Lastly, we can create a table to store our raw data from the csv file before inserting the data into our main tables.

```
CREATE TABLE rawdata
(
   app_name VARCHAR(1000),
   avg_user_rating float,
   user_rating_count int,
   price float,
   developer CHAR(100),
   age_rating VARCHAR(10),
   languages CHAR(10),
   size float,
   primary_genre CHAR(100),
   genres CHAR(100),
   release_date DATE,
   current_v_realease_date DATE

);
```

After creating our database, table and user we can now start creating our main tables as shown in Table 2 below.

CREATE commands	Descriptions
CREATE TABLE language (language_id INT AUTO_INCREMENT, language_type CHAR(10) NOT NULL, PRIMARY KEY(language_id));	Create table called language with attributes language_id (primary key) and language_type
CREATE TABLE developer (developer_id INT AUTO_INCREMENT, developer_name CHAR(100) NOT NULL, PRIMARY KEY(developer_id));	Create table called developer with attributes developer_id (primary key) and developer_name
CREATE TABLE user_rating (user_rating_id INT AUTO_INCREMENT, rating_score FLOAT NOT NULL, PRIMARY KEY(user_rating_id));	Create table called user_rating with attributes user_rating_id (primary key) and rating_score
CREATE TABLE age_rating (age_rating_id INT AUTO_INCREMENT, rating VARCHAR(10) NOT NULL, PRIMARY KEY(age_rating_id));	Create table called age_rating with attributes age_rating_id (primary key) and rating
CREATE TABLE app (app_id INT AUTO_INCREMENT, app_name VARCHAR(1000) NOT NULL, avg_user_count INT NOT NULL, price FLOAT NOT NULL, size FLOAT NOT NULL, developer_id INT NOT NULL, user_rating_id INT NOT NULL, release_date DATETIME NOT NULL, current_v_date DATETIME NOT NULL, age_rating_id INT NOT NULL, PRIMARY KEY(app_id), FOREIGN KEY (developer_id) REFERENCES developer(developer_id), FOREIGN KEY (user_rating_id) REFERENCES user_rating(user_rating_id), FOREIGN KEY (age_rating_id) REFERENCES age_rating(age_rating_id));	Create table called app with attributes app_id (primary key), app_name, avg_user_count, price, size, developer_id (foreign key), user_rating_id (foreign key), release_date, current_v_date and age_rating_id (foreign key)
CREATE TABLE app_genre (app_id INT NOT NULL, primary_genre CHAR(100) NOT NULL, genre CHAR(100) NOT NULL, FOREIGN KEY (app_id) REFERENCES app(app_id));	Create table called app_genre with attributes app_id (foreign key), primary_genre and genre

CREATE TABLE app_language (
  app_id INT NOT NULL,
  language_id INT NOT NULL,
  FOREIGN KEY (app_id) REFERENCES app(app_id),
  FOREIGN KEY (language_id) REFERENCES language(language_id)
);
	Create table called app_language with attributes app_id (foreign key) and language_id(foreign key)

3.2 Enter instance data

After creating the “rawdata” table we then load data from our subset csv file inside:

```
BULK INSERT rawdata

FROM 'C:/Users/Kelly Chin/Documents/Adv Database/mobileapp_1nf.csv'

WITH

(

FIRSTROW = 2, --Second row if header row in file

FIELDTERMINATOR = ',',  --CSV field delimiter

ROWTERMINATOR = '\n',   --Use to shift the control to next row

TABLOCK

)
```

We then add data into our main tables using “rawdata”, as seen in Figure 2 above. We first insert into tables with no foreign key available.

```
INSERT INTO language (language_type)

SELECT DISTINCT languages FROM rawdata;


INSERT INTO developer (developer_name)

SELECT DISTINCT developer FROM rawdata;


INSERT INTO user_rating (rating_score)

SELECT DISTINCT avg_user_rating FROM rawdata;


INSERT INTO age_rating (rating)

SELECT DISTINCT age_rating FROM rawdata;
```

Firstly, our app table containing multiple foreign keys linked many table such as developer and user_rating. Hence, we run the following command to insert data into app table:

```
INSERT INTO app(app_name,avg_user_count,price,size,developer_id,
 user_rating_id, release_date,current_v_date,age_rating_id)
 SELECT
rawdata.app_name,rawdata.user_rating_count,rawdata.price,rawdata.size,developer.developer_id,
 user_rating.user_rating_id,
rawdata.release_date,rawdata.current_v_realease_date,age_rating.age_rating_id
 FROM rawdata
 JOIN developer ON rawdata.developer = developer.developer_name
 JOIN user_rating ON rawdata.avg_user_rating = user_rating.rating_score
 JOIN age_rating ON rawdata.age_rating = age_rating.rating
```

Similarly, to other table with foreign key, we use the same method to insert data into app_genre and app_language:

```
INSERT INTO app_genre(app_id,primary_genre,genre)
 SELECT DISTINCT app.app_id,rawdata.primary_genre,rawdata.genres
 FROM rawdata
 JOIN app
 ON rawdata.app_name = app.app_name
 AND rawdata.user_rating_count = app.avg_user_count
 AND rawdata.price = app.price
 AND rawdata.size = app.size
 AND rawdata.release_date = app.release_date
 AND rawdata.current_v_realease_date = app.current_v_date

INSERT INTO app_language (app_id,language_id)
 SELECT DISTINCT app.app_id,language.language_id
 FROM rawdata
 JOIN app ON rawdata.app_name = app.app_name
 JOIN language ON rawdata.languages = language.language_type
```

3.3 Reflect on how well the database reflects the data

As pre-processing were done in python before inserting the data, hence the data is perfectly fitted into its own tables.

3.4 List SGL commands that answer questions identified in Stage1/Stage 3

Question 1: "What are the most popular genres in the strategy games?"

Selecting and grouping the primary genre and genre to get the sum of each genre with the highest filter rating. The tables shows that the popular genres of high rated mobile application were games, strategy, and entertainment. However, since the data we used its all strategy games hence ignoring both of the genres, we can tell that entertainment, role playing and puzzle is the most popular category in the high rated application. The table give the insights of what the possible user likes and dislike. The genre like music and lifestyle has very little high rated application. Game designer should avoid these genre of games however if he wish to do such genre, more research might be needed so it will be more attractive to user.

```
select p.primary_genre,count(a.app_id) as sum from app a
join app_genre p on a.app_id = p.app_id
join user_rating r on a.user_rating_id = r.user_rating_id
WHERE r.rating_score = 5.0
GROUP by p.primary_genre
UNION
select p.genre,count(a.app_id) as sum from app a
join app_genre p on a.app_id = p.app_id
join user_rating r on a.user_rating_id = r.user_rating_id
WHERE r.rating_score = 5.0
GROUP by p.genre
ORDER BY sum DESC;
```

primary_genre	sum
Games	2241
Strategy	869
Entertainment	510
Role Playing	238
Puzzle	163
Board	125
Simulation	117
Family	95
Action	60
Casual	32
Adventure	14
Card	12
Music	3
Lifestyle	3

14 rows in set (0.01 sec)

Question 2: "Will users be more interested in newly launched game application whenever there is one?"

From the data, we can tell that the higher rated application were the application launched in recent years. However, looking at the average user count column we can see that application launched in 2012 and 2013 has genuinely high user count. This shows that many people will still be attracted to older game application hence the date launched of the application is not so important. We can say that user look more at the features of the mobile game application than its newest of the application.

```
select YEAR(a.release_date),AVG(rating_score),AVG(avg_user_count),count(a.app_id) from app a

join user_rating r on a.user_rating_id = r.user_rating_id

GROUP BY YEAR(a.release_date)

ORDER BY AVG(rating_score) DESC;
```

YEAR(a.release_date)	AVG(rating_score)	AVG(avg_user_count)	count(a.app_id)
2019	4.562274368231047	4660.6968	554
2015	4.520628683693516	30864.5796	509
2013	4.492957746478873	10887.4085	568
2017	4.487107623318385	29493.7567	892
2018	4.4443365695792885	18077.5864	1545
2016	4.357578397212544	34718.4852	1148
2014	4.341386554621849	63998.9055	476
2012	4.324894514767933	405680.3734	474
2010	4.084677419354839	10437.4758	124
2011	3.996173469387755	5115.3546	392
2008	3.9285714285714284	3471.8571	21
2009	3.7153846153846155	15798.0308	65

12 rows in set (0.02 sec)

Question 3: "Is free mobile game application always more popular than paid application?"

From the in_app_purchase column, we can tell that most of the data are free of charge where it is obviously show that there are more 0 than 1. We can tell that people tend to go for the free app rather than the paid app, as the target user of the strategy games are mostly youngster. This means that they might not willing to spend money on paid app hence result in free mobile game application are more popular. From the graph, we can tell that age rating of above 17 have relatively lesser data, as working adults might not have leisure time to go for game application, hence designing a game application that is suitable for below 17 are quite important.

count(a.app_id)	in_app_purchase	rating_score	rating
48	0	3	9+
66	0	3	17+
2	0	3	12+
106	0	3.5	9+
39	0	3.5	4+
66	0	3.5	17+
2	0	3.5	12+
191	0	4	4+
649	0	4	12+
17	0	4	17+
518	0	4	9+
1238	0	4.5	4+
1282	0	4.5	9+
1214	0	4.5	12+
104	0	4.5	17+
232	0	5	9+
333	0	5	4+
252	0	5	12+
22	0	5	17+
2	1	3	9+
51	1	4	4+
32	1	4	12+
12	1	4	9+
30	1	4.5	4+
109	1	4.5	12+
95	1	4.5	9+
26	1	4.5	17+
30	1	5	9+

```
select count(a.app_id),a.price as in_app_purchase,r.rating_score,g.rating from app a join user_rating r
on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price = 0 AND r.rating_score = 3.0 group by g.age_rating_id
UNION
select count(a.app_id),a.price as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price = 0 AND r.rating_score = 3.5 group by g.age_rating_id
UNION
select count(a.app_id),a.price as in_app_purchase,r.rating_score,g.rating from app a join user_rating r
on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price = 0 AND r.rating_score = 4.0 group by g.age_rating_id
UNION
select count(a.app_id),a.price as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price = 0 AND r.rating_score = 4.5 group by g.age_rating_id
UNION
select count(a.app_id),a.price as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price = 0 AND r.rating_score = 5.0 group by g.age_rating_id
UNION
select count(a.app_id),1 as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price > 0 AND r.rating_score = 3.0 group by g.age_rating_id
UNION
select count(a.app_id),1 as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price > 0 AND r.rating_score = 3.5 group by g.age_rating_id
UNION
select count(a.app_id),1 as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price > 0 AND r.rating_score = 4.0 group by g.age_rating_id
UNION
select count(a.app_id),1 as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price > 0 AND r.rating_score = 4.5 group by g.age_rating_id
UNION
select count(a.app_id),1 as in_app_purchase,r.rating_score,g.rating from app a
join user_rating r on a.user_rating_id = r.user_rating_id join age_rating g on a.age_rating_id = g.age_rating_id
where a.price > 0 AND r.rating_score = 5.0 group by g.age_rating_id
order by count(a.app_id)
```

4 Create simple web application

Using the web application, users can choose from different factors to view the different available to the chosen data. From the filter table they can get the insights of what type of data are there in the chosen filtered features. The motivation behind the web application is to provide an easier searching and viewing the data in each category of the different features.

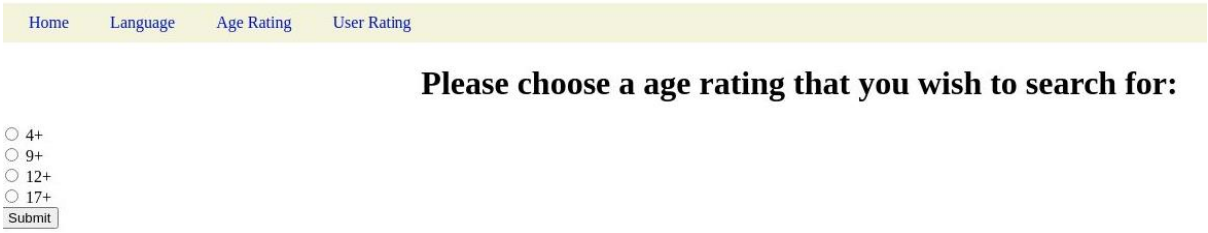
To generate the web application, the user has to enter “node index.js” in the terminal to create a connection with host, user, password and the database “mobileApp” we created, and run-on port 8808. We will be querying our database using index.js and our web application will be made using express.

```
const express = require('express');
const mustacheExpress = require('mustache-express');
let bodyParser = require('body-parser');
const app = express();
const webPort = 8088;
const path = require('path');
app.set('views', `${__dirname}/views`);
app.set('view engine', 'mustache');
app.engine('mustache', mustacheExpress());
app.engine('html', mustacheExpress());
app.use(bodyParser.urlencoded({ extended: true }));
app.listen(webPort, function () {
    console.log("Server started , listen to " + webPort);
});
const mysql = require('mysql');
const db = mysql.createConnection(
    {
        host: 'localhost',
        user: 'kelly',
        password: 'sqlpw',
        database: 'mobileApp'
    }
)

db.connect((err) => {
    if (err) {
        print(err);
        throw err;
    }
    console.log("Connected to database");
});
```

For every main page there are different factors for user to choose their data accordingly.

When user click on the age rating tab, it will allow user to choose the category.



This is done by creating the different age rating in a form. After the user click on the submit button it will pass the data to the /ar-filter.

```
<body>
    <ul>
        <li><a href='/'>Home</a></li>
        <li><a href='/lang'>Language</a></li>
        <li><a href='/ar'>Age Rating</a></li>
        <li><a href='/ur'>User Rating</a></li>
    </ul>

    <h1>Please choose a age rating that you wish to search for:</h1>

    <form action="/ar-filter">
        <input type="radio" id="ar_y1" name="ar_y" value="4+">
        <label for="ar_y1">4+</label><br>
        <input type="radio" id="ar_y2" name="ar_y" value="9+">
        <label for="ar_y2">9+</label><br>
        <input type="radio" id="ar_y3" name="ar_y" value="12+">
        <label for="ar_y3">12+</label><br>
        <input type="radio" id="ar_y4" name="ar_y" value="17+">
        <label for="ar_y4">17+</label><br>
        <input type="submit" value="Submit">
    </form>
</body>
```

Below show the table of 12+ age rating after selecting it. The table provide the name, price, rating and genre of the data in this category.

Back

The mobile app with 12+ age rating.

Name	Price	Rating	Genre
Chain Strikeu2122	0	4	Games
Name	**Price**	**Rating**	**Genre**
HEIR OF LIGHT	0	4	Games
Name	**Price**	**Rating**	**Genre**
Tokyo Ghoul Dark War	0	4	Games
Name	**Price**	**Rating**	**Genre**
Final Fantasy XV A New Empire	0	4	Games

This is done by putting the selected value into query and join the necessary table such as age_rating table then filter out the data where it is the same as the selected value.

```
app.get('/ar-filter', function (req, res) {
    fAR = req.query.ar_y
    var query = "select distinct app.app_name,app.price,user_rating.rating_score,app_genre.prima
    var t = fAR
    db.query(query, (err, result) => {
        if (err) {
            res.redirect("/");

        }
        res.render('age-rt-filter.html',
            {
                data: result,
                t: t
            })
    })

})
```

Every factors works similarly as the quote above.

Please choose a language that you wish to search for:

- EN
- FR
- DE
- JA
- KO
- ZH

Submit

Back

The mobile app with language JA.

Name	Price	Rating	Age Rating
Chess tChess Pro	7.99	4	4+
Name	**Price**	**Rating**	**Age Rating**
UniWar Multiplayer Strategy	0	3.5	9+
Name	**Price**	**Rating**	**Age Rating**
Pixel Starshipsu2122	0	4.5	9+
Name	**Price**	**Rating**	**Age Rating**
u25bbCHESS	0	4.5	4+

5 References

Tristan (2019) 17K mobile strategy games, Kaggle. Available at: https://www.kaggle.com/datasets/tristan581/17k-apple-app-store-strategy-games (Accessed: January 17, 2023).

My shareable lab link:

https://hub.labs.coursera.org:443/connect/sharedqseyodgs?forceRefresh=false&path=%2F%3Ffolder%3D%2Fhome%2Fcoder%2Fproject