

Project Report

About

Database project

AppMySeries

Submitted by

Miss. Sasithon Dontree 6413050001 Response 33.3%

Miss. Jutara Rungsook 6413050011 Response 33.3%

Miss. Thannicha sompan 6413050038 Response 33.3%

Present

Ms. Sunisa Sathapornvajana

INT205 Database management system

1/2565

Introduction

รายงานเล่มนี้ จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของรายวิชา INT205 Database management system โดยภายในเล่มรายงานนั้นเป็นการอธิบายเกี่ยวกับรายละเอียดการออกแบบฐานข้อมูลเพื่อการนำไปใช้ของ Application ที่เกี่ยวกับการใช้เข้ารับชมซีรีส์ชื่อว่า AppMySeries ซึ่งภายใน Application จะเป็นการสมัครเพื่อใช้งาน ภายในแอปพลิเคชัน มีทั้งซีรีส์ไทยและต่างประเทศ ใช้งานง่าย เป็นระบบที่คุ้นเคย

เหตุผลของการจัดทำฐานเก็บข้อมูลหัวข้อดังกล่าวนี้ เนื่องจากทางคณะผู้จัดทำมีความสนใจเกี่ยวกับการรับชมซีรีส์ และได้หวังว่ารายงานเล่มนี้จะเป็นประโยชน์ต่อผู้อ่าน ผู้ที่ต้องการทราบถึงโครงสร้างฐานข้อมูลของการทำ Application เกี่ยวกับซีรีส์ หรือผู้ที่ต้องการศึกษาเกี่ยวกับการสร้างฐานข้อมูลแบบพื้นฐาน ทำให้คณะผู้จัดทำเลือกทำการออกแบบฐานข้อมูลของ Application ดังกล่าวมาเป็นเนื้อหาของรายงานฉบับนี้ หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับและขออภัยมา ณ ที่นี้ด้วย

Table of Content

Subject	Page
Introduction	1
Table of Content (TOC)	2
Business Requirements	3
Logical DB Design (ER Diagram)	4
SQL Statements	5
Data Dictionary	10
DDL script for creating the database	13
Export Data in the format of INSERT statements	21

Business Requirements

User View ของ AppMySeries เป็นมุมมองของผู้ใช้ขณะที่เข้าใช้ ซึ่งแอปพลิเคชันของกลุ่มเรามีเนื้อหาเกี่ยวกับการเข้าใช้เพื่อรับชมซีรีส์ โดยเมื่อ user เข้าไปที่หน้าแรกของแอปพลิเคชัน จะมีการให้สมัครเข้าใช้งานเพื่อเข้าสู่ระบบ และเมื่อสมัครเสร็จผู้ใช้จะสามารถเข้าชมซีรีส์ทั้งไทยและต่างประเทศได้ทั้งหมดที่มีภายในแอปพลิเคชัน มีระบบการสมัครสมาชิก Package VIP เพื่อเพิ่มสิทธิประโยชน์ในการรับชม โดยสามารถรับชมได้คมชัดขึ้นและสามารถเซฟวิดีโอเก็บไว้ดูได้เพื่อเพิ่มอรรถรสในการรับชม เป็นในรูปแบบรายเดือน

Requirements

User Table: เก็บรวบรวมข้อมูลของ user ตอนสมัครเข้าใช้งานแอปพลิเคชัน

User_Watch Table: เก็บประวัติการเข้าชม series ในแต่ละครั้งว่ารับชม series เรื่องอะไร ตอนที่เท่าไร

Series Table: เก็บข้อมูลเกี่ยวกับซีรีส์ ประเภทของซีรีส์ (Type) จำนวนตอน (episode) ความนิยม (Rating)

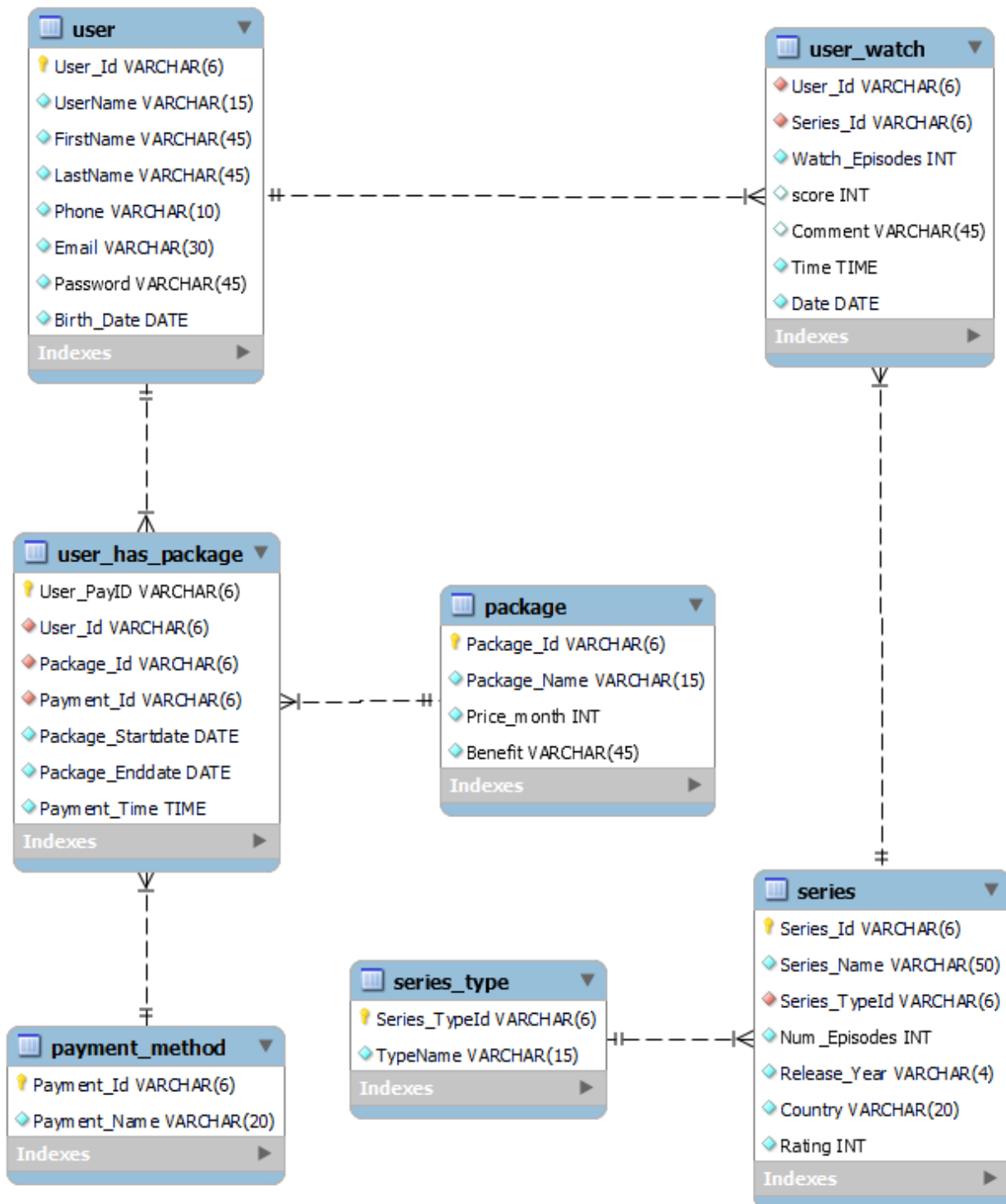
Series_Type Table: เก็บข้อมูลเกี่ยวกับประเภทของซีรีส์ที่อาจมีเพิ่มเข้ามาอีกในอนาคตจึงแยกย่อยออกมาจาก Series Table

User_has_package Table: เก็บข้อมูลเกี่ยวกับประวัติการสมัครแพ็คเกจของผู้ใช้ ประเภทของแพ็คเกจที่ผู้ใช้สมัคร ช่องทางการจ่ายเงินที่ผู้ใช้เลือกชำระเงิน

Package Table: เก็บข้อมูลเกี่ยวกับแพ็คเกจที่มีในแอปพลิเคชัน ชื่อแพ็คเกจ ราคาของแพ็คเกจ สิทธิประโยชน์ที่ได้ภายในแพ็คเกจ

Payment_method Table: เก็บรายละเอียดเกี่ยวกับวิธีการชำระเงินในการสมัครแพ็คเกจ

Logical DB Design (ER Diagram)



SQL Statements

1. Join table

โจทย์ : แสดงข้อมูลผู้ใช้ที่ซื้อแพ็คเกจราคา 500 บาท โดยให้แสดงรหัสผู้ใช้และชื่อแพ็คเกจร่วมด้วยเพื่อเช็คความถูกต้อง

```
select u.User_Id,UserName,Package_Name,Price_Month
from user u join user_has_package up
on u.User_Id = up.User_Id
join package p on up.Package_Id = p.Package_Id
where Price_Month = 500;
```

Query5 ข้อ 1

```
-- 1.join
-- ต้องการดูว่าใครซื้อ package ราคา 500
select u.User_Id,UserName,Package_Name,Price_Month
from user u join user_has_package up
on u.User_Id = up.User_Id
join package p on up.Package_Id = p.Package_Id
where Price_Month = 500;
```

User_Id	UserName	Package_Name	Price_Month
US0002	Ni	VIP Premium	500
US0003	Meepoo	VIP Premium	500
US0001	Aon	VIP Premium	500
US0005	Vin	VIP Premium	500
US0007	Bang	VIP Premium	500

Result 14

#	Time	Action	Message
67	13:19:31	select year(Date),avg(score) as Rating_Avg_Year from user_watch where series_Id ='SR00...	1 row(s) returned
68	13:20:51	create or replace view User_member as select User_Id,FirstName,Package_Id from user u j...	Error Code: 1052. Column 'User_Id' in field list is ambiguous
69	13:21:09	create or replace view User_member as select u.User_Id,FirstName,Package_Id from user ...	0 row(s) affected
70	13:21:33	create or replace view User_member as select u.User_Id,FirstName,Package_Id from user ...	0 row(s) affected
71	13:21:41	select * from User_member LIMIT 0, 5000	10 row(s) returned
72	13:23:32	select u.User_Id,UserName,Package_Name,Price_Month from user u join user_has_packa...	5 row(s) returned

2. Subquery + join

โจทย์ : แสดงข้อมูลของผู้ใช้ที่สมัครแพ็คเกจที่ได้รับสิทธิประโยชน์คือ '1089P'

```
select up.User_Id,u.UserName,up.Package_Startdate
```

```
from user_has_package up join user u
```

```
on u.User_Id = up.User_Id
```

```
where Package_Id = (select Package_Id
```

```
from package
```

```
where Benefit = '1089P');
```

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

```

8      where Price_Month = 500;
9
10     -- 2.subquery+join
11     --หาuserที่มีpackageที่มีbenefit คือ 1089P
12     • select up.User_Id,u.UserName,up.Package_Startdate
13         from user_has_package up join user u
14         on u.User_Id = up.User_Id
15     where Package_Id = (select Package_Id
16                         from package
17                         where Benefit = '1089P');
18

```

The results pane shows a table with the following data:

User_Id	UserName	Package_Startdate
US0003	Meepoo	2022-05-15
US0002	Ni	2022-09-01
US0004	Kan	2022-08-23
US0006	Kan	2022-05-12
US0008	Pam	2022-01-31

Below the results table, there is an 'Action Output' section showing the execution of the query:

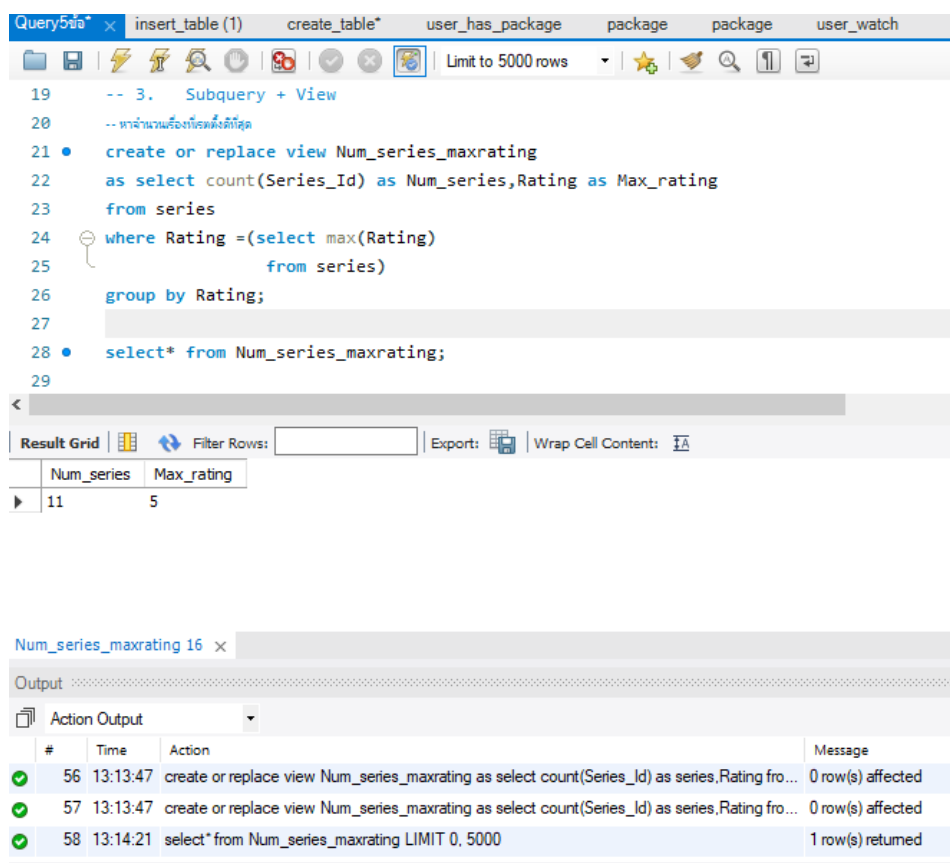
#	Time	Action	Message
72	13:23:32	select u.User_Id,UserName,Package_Name,Price_Month from user u join user_has_packa...	5 row(s) returned
73	13:27:06	select up.User_Id,u.UserName,up.Package_Startdate from user_has_package up join user ...	5 row(s) returned

3. Subquery + View

โจทย์ : หาจำนวนเรื่องที่คะแนนความนิยมดีที่สุด

```
create or replace view Num_series_maxrating
as select count (Series_Id) as Num_series,Rating as Max_rating
from series
where Rating = (select max (Rating)
                from series)
group by Rating;

select* from Num_series_maxrating;
```



The screenshot shows a SQL IDE with a script editor and a results/output pane. The script editor contains the following SQL code:

```
-- 3. Subquery + View
-- หาจำนวนเรื่องที่คะแนนดีที่สุดในที่สุด
create or replace view Num_series_maxrating
as select count(Series_Id) as Num_series,Rating as Max_rating
from series
where Rating =(select max(Rating)
               from series)
group by Rating;
select* from Num_series_maxrating;
```

The results pane shows the output of the SQL script. It includes a table with the results of the final query and an action log.

Num_series	Max_rating
11	5

#	Time	Action	Message
56	13:13:47	create or replace view Num_series_maxrating as select count(Series_Id) as series,Rating fro...	0 row(s) affected
57	13:13:47	create or replace view Num_series_maxrating as select count(Series_Id) as series,Rating fro...	0 row(s) affected
58	13:14:21	select* from Num_series_maxrating LIMIT 0, 5000	1 row(s) returned

4. aggregate functions

โจทย์: หาค่าเฉลี่ยคะแนนการรับชมของซีรีส์เรื่อง 'Kingdom 1' ทั่วประเทศ

```
select uw.series_Id,series_Name,year(Date),avg(score) as Rating_Avg_Year
from user_watch uw join series s on s.series_Id = uw.series_Id
where s.series_Name = 'Kingdom 1'
group by year (Date);
```

Query5 หน้า x insert_table (1) create_table* user_has_package package package user_watch

Limit to 5000 rows

```

29
30 -- 4.aggregate functions
31 -- หาค่าเฉลี่ยของscoreแบบรายปี ของseries เรื่อง kingdom1
32 • select uw.series_Id,series_Name,year(Date),avg(score) as Rating_Avg_Year
33   from user_watch uw join series s on s.series_Id = uw.series_Id
34   where s.series_Name = 'Kingdom 1'
35   group by year(Date);
36

```

Result Grid

series_Id	series_Name	year(Date)	Rating_Avg_Year
SR0010	Kingdom 1	2022	5.0000

Result 31 x

Output

Action Output

#	Time	Action	Message
✓ 88	15:25:19	select uw.series_Id,series_Name,year(Date),avg(score) as Rating_Avg_Year from user_wat...	1 row(s) returned
✓ 89	15:25:49	select* from Num_series_maxrating LIMIT 0, 5000	1 row(s) returned
✓ 90	15:25:52	select uw.series_Id,series_Name,year(Date),avg(score) as Rating_Avg_Year from user_wat...	1 row(s) returned

5. view + join

โจทย์ : package Id ของผู้ใช้แต่ละคนอะไร

create or replace view User_member

as select u.User_Id,FirstName,Package_Id

from user u join user_has_package up

on u.User_Id = up.User_Id;

select *from User_member;

The screenshot shows a SQL IDE interface with a query window titled 'Query5' containing the following SQL code:

```

37  -- 5.view+join
38  -- userคนไหนใช้package Idอะไร
39  • create or replace view User_member
40  as select u.User_Id,FirstName,Package_Id
41  from user u join user_has_package up
42  on u.User_Id = up.User_Id;
43
44  • select *from User_member;

```

Below the query window, the 'Result Grid' displays the output of the query. It shows 10 rows of data with columns: User_Id, FirstName, and Package_Id.

User_Id	FirstName	Package_Id
US0002	Thannicha	PK0002
US0003	Sasithon	PK0001
US0003	Sasithon	PK0002
US0002	Thannicha	PK0001
US0001	Jutara	PK0002
US0004	Kamolsin	PK0001
US0005	Nuttayaporn	PK0002
US0006	Danaithep	PK0001
US0007	Thanasak	PK0002

Below the result grid, the 'Action Output' window shows the execution log:

#	Time	Action	Message
75	13:34:52	select year(Date),avg(score) as Rating_Avg_Year from user_watch where series_Id ='SR00...	1 row(s) returned
76	13:35:42	select series_Id,year(Date),avg(score) as Rating_Avg_Year from user_watch where series_I...	1 row(s) returned
77	13:37:52	select * from User_member LIMIT 0, 5000	10 row(s) returned

Data Dictionary

Table: User

Description: เก็บข้อมูลต่างๆของผู้ใช้ในการสมัคร/เข้าสู่ระบบ

No	Column Name	Description	Data Type	Constraint	Referenced Table
1	User_Id	รหัสผู้ใช้	VARCHAR (6)	PK	
2	UserName	ชื่อที่ใช้ในแอป ฯ	VARCHAR (15)	NN	
3	FirstName	ชื่อจริง	VARCHAR (45)	NN	
4	LastName	นามสกุล	VARCHAR (45)	NN	
5	Phone	เบอร์โทร	VARCHAR (10)	NN	
6	Email	อีเมลที่ใช้สมัคร	VARCHAR (30)	NN	
7	Password	รหัสผ่าน	VARCHAR (45)	NN, UK	
8	Birth_Date	วัน/เดือน/ปีเกิด	DATE	NN	

Table: User_watch

Description: ตารางแสดงข้อมูลว่าผู้ใช้ดูซีรีส์เรื่องอะไรและให้เรตติ้งแต่ละตอนเท่าไร

No	Column Name	Description	Data Type	Constraint	Referenced Table
1	User_Id	รหัสผู้ใช้	VARCHAR (6)	FK	User
2	Series_id	รหัสซีรีส์	VARCHAR (6)	FK	Series
3	Watch_Episode	ตอนที่กำลังดู	INT	NN	
4	Score	คะแนนของ EP	INT		
5	Comment	ความคิดเห็น	VARCHAR (45)		
6	Time	เวลาที่เข้าดูซีรีส์	TIME	NN	
7	Date	วันที่เข้าดูซีรีส์	DATE	NN	

Table : Series

Description: ตารางแสดงข้อมูลซีรีส์

No	Column Name	Description	Data Type	Constraint	Referenced Table
1	Series_id	รหัสซีรีส์	VARCHAR (6)	PK	
2	Series_Name	ชื่อซีรีส์	VARCHAR (50)	NN	
3	Series_TypeId	รหัสประเภทซีรีส์	VARCHAR (6)	FK	Series_type
4	Num_Episode	จำนวนEP ทั้งหมด	INT	NN	
5	Release_Year	ปีแรกที่ซีรีส์ลงจอ	VARCHAR (4)	NN	
6	Country	ประเทศที่มาซีรีส์	VARCHAR (20)	NN	
7	Rating	คะแนนเฉลี่ยแต่ละ EP	INT	NN	

Table : Series_type

Description: ตารางแสดงข้อมูลประเภทของซีรีส์

No	Column Name	Description	Data Type	Constraint	Referenced Table
1	Series_TypeId	รหัสประเภทซีรีส์	VARCHAR (6)	PK	
2	TypeName	ชื่อประเภทซีรีส์	VARCHAR (15)	NN	

Table : User_has_package

Description: ตารางแสดงข้อมูลว่าผู้ใช้สมัครแพ็คเกจอะไร เริ่มสมัครและสิ้นสุดวันไหน

No	Column Name	Description	Data Type	Constraint	Referenced Table
1	User_PayId	รหัสผู้จ่าย	VARCHAR (6)	PK	
2	User_Id	รหัสผู้ใช้	VARCHAR (6)	FK	User
3	Package_Id	แพ็คเกจ	VARCHAR (6)	FK	package
4	Payment_Id	รหัส Payment	VARCHAR (6)	FK	Payment_method
5	Package_Startdate	วันเริ่มสมัคร	DATE	NN	
6	Package_Enddate	วันหมดอายุ	DATE	NN	
7	Payment_Time	เวลาในการจ่าย	TIME	NN	

Table: Payment_method

Description: ตารางแสดงข้อมูลช่องทางการจ่ายเงินค่าสมัครแพ็คเกจ

No	Column Name	Description	Data Type	Constraint	Referenced Table
1	Payment_Id	รหัส Payment	VARCHAR (6)	PK	
2	Payment_Name	Payment ที่ใช้	VARCHAR (20)	NN	

Table: Package

Description: ตารางแสดงข้อมูลแพ็คเกจที่เปิดให้สมัคร

No	Column Name	Description	Data Type	Constraint	Referenced Table
1	Package_Id	รหัสแพ็คเกจ	VARCHAR (6)	PK	
2	Package_Name	ชื่อแพ็คเกจ	VARCHAR (15)	NN	
3	Price_month	ราคาแพ็คเกจ	INT	NN	
4	Benefit	สิทธิประโยชน์	VARCHAR (45)	NN	

DDL script for creating the database

```
-- DROP DATABASE appmyseries ;
```

```
-- MySQL Script generated by MySQL Workbench
```

```
-- Mon Nov 28 12:09:34 2022
```

```
-- Model: New Model   Version: 1.0
```

```
-- MySQL Workbench Forward Engineering
```

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
```

```
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
```

```
SET @OLD_SQL_MODE=@@SQL_MODE,
```

```
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_
DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
-----
```

```
-- Schema AppMySeries
```

```
-----
```

```
-----
```

```
-- Schema AppMySeries
```

```
-----
```

```
CREATE SCHEMA IF NOT EXISTS `AppMySeries` DEFAULT CHARACTER SET utf8 ;
```

```
USE `AppMySeries` ;
```

```
-----
```

```
-- Table `AppMySeries`.`User`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `AppMySeries`.`User` (
```

```
  `User_Id` VARCHAR(6) NOT NULL,
```

```
  `UserName` VARCHAR(15) NOT NULL,
```

```
  `FirstName` VARCHAR(45) NOT NULL,
```

```
  `LastName` VARCHAR(45) NOT NULL,
```

```
  `Phone` VARCHAR(10) NOT NULL,
```

```
  `Email` VARCHAR(30) NOT NULL,
```

```
  `Password` VARCHAR(45) NOT NULL,
```

```
  `Birth_Date` DATE NOT NULL,
```

```
  PRIMARY KEY (`User_Id`),
```

```
  UNIQUE INDEX `password_UNIQUE` (`Password` ASC) VISIBLE)
```

```
ENGINE = InnoDB;
```

```
-----
```

```
-- Table `AppMySeries`.`Series_type`
```

```
-----
CREATE TABLE IF NOT EXISTS `AppMySeries`.`Series_type` (
```

```
  `Series_TypeId` VARCHAR(6) NOT NULL,
```

```
  `TypeName` VARCHAR(15) NOT NULL,
```

```
  PRIMARY KEY (`Series_TypeId`))
```

```
ENGINE = InnoDB;
```

```
-----
-- Table `AppMySeries`.`Series`
-----
```

```
CREATE TABLE IF NOT EXISTS `AppMySeries`.`Series` (
```

```
  `Series_Id` VARCHAR(6) NOT NULL,
```

```
  `Series_Name` VARCHAR(50) NOT NULL,
```

```
  `Series_TypeId` VARCHAR(6) NOT NULL,
```

```
  `Num_Episodes` INT NOT NULL,
```

```
  `Release_Year` VARCHAR(4) NOT NULL,
```

```
  `Country` VARCHAR(20) NOT NULL,
```

```
  `Rating` INT NOT NULL,
```

```
  PRIMARY KEY (`Series_Id`),
```

```
  UNIQUE INDEX `Movie_Id_UNIQUE` (`Series_Id` ASC) VISIBLE,
```

```
  INDEX `fk_Movie_Movie_type1_idx` (`Series_TypeId` ASC) VISIBLE,
```



```

CONSTRAINT `fk_Movie_Movie_type1`

FOREIGN KEY (`Series_TypeId`)

REFERENCES `AppMySeries`.`Series_type` (`Series_TypeId`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-----

-- Table `AppMySeries`.`Package`

-----

CREATE TABLE IF NOT EXISTS `AppMySeries`.`Package` (

  `Package_Id` VARCHAR(6) NOT NULL,

  `Package_Name` VARCHAR(15) NOT NULL,

  `Price_month` INT NOT NULL,

  `Benefit` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`Package_Id`),

  UNIQUE INDEX `Package_Id_UNIQUE` (`Package_Id` ASC) VISIBLE)

ENGINE = InnoDB;

-----

-- Table `AppMySeries`.`User_watch`

```

```

-----

CREATE TABLE IF NOT EXISTS `AppMySeries`.`User_watch` (

  `User_Id` VARCHAR(6) NOT NULL,

  `Series_Id` VARCHAR(6) NOT NULL,

  `Watch_Episodes` INT NOT NULL,

  `score` INT NULL,

  `Comment` VARCHAR(45) NULL,

  `Time` TIME NOT NULL,

  `Date` DATE NOT NULL,

  INDEX `fk_User_copy1_has_Movie_copy1_Movie_copy11_idx` (`Series_Id` ASC) VISIBLE,

  INDEX `fk_User_copy1_has_Movie_copy1_User_copy11_idx` (`User_Id` ASC) VISIBLE,

  CONSTRAINT `fk_User_copy1_has_Movie_copy1_User_copy11`

    FOREIGN KEY (`User_Id`)

      REFERENCES `AppMySeries`.`User` (`User_Id`)

      ON DELETE NO ACTION

      ON UPDATE NO ACTION,

  CONSTRAINT `fk_User_copy1_has_Movie_copy1_Movie_copy11`

    FOREIGN KEY (`Series_Id`)

      REFERENCES `AppMySeries`.`Series` (`Series_Id`)

      ON DELETE NO ACTION

```

ON UPDATE NO ACTION)

ENGINE = InnoDB;

```
-----
-- Table `AppMySeries`.`Payment_method`
-----
```

```
CREATE TABLE IF NOT EXISTS `AppMySeries`.`Payment_method` (

  `Payment_Id` VARCHAR(6) NOT NULL,

  `Payment_Name` VARCHAR(20) NOT NULL,

  PRIMARY KEY (`Payment_Id`),

  UNIQUE INDEX `Package_Id_UNIQUE` (`Payment_Id` ASC) VISIBLE)

ENGINE = InnoDB;
```

```
-----
-- Table `AppMySeries`.`User_has_package`
-----
```

```
CREATE TABLE IF NOT EXISTS `AppMySeries`.`User_has_package` (

  `User_PayID` VARCHAR(6) NOT NULL,

  `User_Id` VARCHAR(6) NOT NULL,

  `Package_Id` VARCHAR(6) NOT NULL,
```

```

`Payment_Id` VARCHAR(6) NOT NULL,

`Package_Startdate` DATE NOT NULL,

`Package_Enddate` DATE NOT NULL,

`Payment_Time` TIME NOT NULL,

INDEX `fk_User_has_Package_Package1_idx` (`Package_Id` ASC) VISIBLE,

INDEX `fk_User_has_Package_User1_idx` (`User_Id` ASC) VISIBLE,

PRIMARY KEY (`User_PayID`),

INDEX `fk_User_has_package_Payment_method1_idx` (`Payment_Id` ASC) VISIBLE,

CONSTRAINT `fk_User_has_Package_User1`

    FOREIGN KEY (`User_Id`)

    REFERENCES `AppMySeries`.`User` (`User_Id`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

CONSTRAINT `fk_User_has_Package_Package1`

    FOREIGN KEY (`Package_Id`)

    REFERENCES `AppMySeries`.`Package` (`Package_Id`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

CONSTRAINT `fk_User_has_package_Payment_method1`

    FOREIGN KEY (`Payment_Id`)

```

```
REFERENCES `AppMySeries`.`Payment_method` (`Payment_Id`)
```

```
ON DELETE NO ACTION
```

```
ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;
```

```
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
```

```
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Export Data in the format of INSERT statements

use appmyseries;

1- user

INSERT INTO user (User_Id, UserName, FirstName, LastName, Phone, Email, Password, Birth_Date)

VALUES

```
(US0001', 'Aon', 'Jutara', 'Rungsook', '0123456789', 'Jutarat@gmail.com', 'PA0001', '2003-01-21'),

(US0002', 'Ni', 'Thannicha', 'Sompan', '0987654321', 'Thannicha@gmail.com', 'PA0002', '2002-03-12'),

(US0003', 'Meepoo', 'Sasithon', 'Dontree', '0143216547', 'Sasithon@gmail.com', 'PA0003', '2003-12-01'),

(US0004', 'Kan', 'Kamolsin', 'Teeyapun', '0234567890', 'Kamolsin@gmail.com', 'PA0004', '2002-12-22'),

(US0005', 'Vin', 'Nuttayaporn', 'Yodsan', '0143216771', 'Nuttayaporn@gmail.com', 'PA0005', '2003-05-15'),

(US0006', 'Kan', 'Danaithep', 'Limskul', '0234567891', 'Danaithep@gmail.com', 'PA0006', '2002-02-22'),

(US0007', 'Bang', 'Thanasak ', 'Yakumpor', '0143216547', 'Thanasak@gmail.com', 'PA0007', '2003-05-16'),

(US0008', 'Pam', 'Pacharapo', 'Udomkiat', '0143216547', 'SPacharapo@gmail.com', 'PA0008', '2002-06-27'),

(US0009', 'Kan', 'Witthawat', 'Boonoi', '0500000105', 'Witthawat@gmail.com', 'PA0009', '2002-03-06'),
```

('US0010', 'Ice', 'Sivilai', 'Whyiantong', '0321554260', 'Sivilai@gmail.com', 'PA0010', '2001-01-08');

	User_Id	UserName	FirstName	LastName	Phone	Email	Password	Birth_Date
▶	US0001	Aon	Jutara	Rungsook	0123456789	Jutarat@gmail.com	PA0001	2003-01-21
	US0002	Ni	Thannicha	Sompan	0987654321	Thannicha@gmail.com	PA0002	2002-03-12
	US0003	Meepoo	Sasithon	Dontree	0143216547	Sasithon@gmail.com	PA0003	2003-12-01
	US0004	Kan	Kamolsin	Teeyapun	0234567890	Kamolsin@gmail.com	PA0004	2002-12-22
	US0005	Vin	Nuttayaporn	Yodsan	0143216771	Nuttayaporn@gmail.com	PA0005	2003-05-15
	US0006	Kan	Danaithep	Limskul	0234567891	Danaithep@gmail.com	PA0006	2002-02-22
	US0007	Bang	Thanasak	Yakumpor	0143216547	Thanasak@gmail.com	PA0007	2003-05-16
	US0008	Pam	Pacharapo	Udomkiat	0143216547	SPacharapo@gmail.com	PA0008	2002-06-27
	US0009	Kan	Witthawat	Boonoi	0500000105	Witthawat@gmail.com	PA0009	2002-03-06
	US0010	Ice	Sivilai	Whyiantong	0321554260	Sivilai@gmail.com	PA0010	2001-01-08
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2- user_watch

INSERT INTO user_watch (User_Id, Series_Id, Watch_Episodes, score, Comment, Time, Date) VALUES

('US0001', 'SR0016', 7, 5, NULL, '00:22:20', '2022-04-14'),

('US0002', 'SR0010', 1, NULL, 'Help', '00:12:30', '2022-06-15'),

('US0002', 'SR0010', 2, NULL, 'What?', '00:13:30', '2022-06-15'),

('US0002', 'SR0010', 3, 5, 'I love her', '00:14:30', '2022-06-15'),

('US0002', 'SR0010', 4, NULL, 'I hate her', '00:15:30', '2022-06-15'),

('US0002', 'SR0010', 5, NULL, '5555', '00:16:30', '2022-06-15'),

('US0002', 'SR0010', 6, 5, 'I love it', '00:17:30', '2022-06-15'),

('US0002', 'SR0009', 1, NULL, Null, '00:18:30', '2022-06-15'),

('US0002', 'SR0009', 2, NULL, NULL, '00:19:30', '2022-06-15'),

('US0002', 'SR0009', 3, NULL, NULL, '00:20:30', '2022-06-15'),

('US0002', 'SR0009', 4, NULL, NULL, '00:21:30', '2022-06-15'),

('US0002', 'SR0009', 5, NULL, NULL, '00:22:30', '2022-06-15'),

('US0002', 'SR0009', 6, 5, 'I love it', '00:23:30', '2022-06-15'),

('US0003', 'SR0022', 9, NULL, NULL, '00:04:00', '2022-01-22'),

('US0003', 'SR0017', 1, 5, '5555 ', '00:04:00', '2022-03-04'),

('US0003', 'SR0024', 5, NULL, NULL, '00:04:00', '2022-12-09'),

('US0003', 'SR0007', 11, NULL, 'What is it?', '00:04:00', '2022-07-25'),

('US0004', 'SR0001', 1, NULL, NULL, '00:01:30', '2022-04-20'),

('US0004', 'SR0001', 2, NULL, NULL, '00:03:30', '2022-05-01'),

('US0005', 'SR0001', 1, 5, 'Why', '00:08:08', '2022-04-20'),

('US0005', 'SR0001', 4, 5, NULL, '00:09:30', '2022-04-20'),

('US0005', 'SR0001', 9, 5, NULL, '00:10:15', '2022-04-20'),

('US0005', 'SR0001', 10, 5, NULL, '00:00:03', '2022-04-21'),

('US0005', 'SR0001', 22, 5, 'i love him', '00:22:00', '2022-04-21'),

('US0005', 'SR0001', 49, 5, NULL, '00:22:50', '2022-04-21'),

('US0006', 'SR0025', 4, NULL, NULL, '00:10:30', '2022-04-18'),

('US0006', 'SR0025', 5, 5, 'Cool', '00:11:30', '2022-04-18'),
('US0006', 'SR0011', 1, NULL, NULL, '00:00:08', '2022-06-20'),
('US0006', 'SR0011', 2, NULL, NULL, '00:01:32', '2022-06-20'),
('US0006', 'SR0011', 3, 4, 'Sweet', '00:02:30', '2022-06-21'),

('US0007', 'SR0022', 3, 5, 'Very Good', '00:01:00', '2022-11-15'),

('US0008', 'SR0003', 1, 4, 'Good', '00:09:00', '2022-06-04'),
('US0008', 'SR0003', 24, 4, 'I love him', '00:09:00', '2022-06-04'),

('US0009', 'SR0017', 1, 5, 'Good', '00:15:30', '2022-08-04'),
('US0009', 'SR0017', 20, 4, 'It OK', '00:16:00', '2022-08-04'),
('US0009', 'SR0017', 34, 3, NULL, '00:16:20', '2022-08-04'),

('US0010', 'SR0018', 38, 5, 'Sweet', '00:14:00', '2022-01-13');

	User_Id	Series_Id	Watch_Episodes	score	Comment	Time	Date
▶	US0001	SR0016	7	5	NULL	00:22:20	2022-04-14
	US0002	SR0010	1	NULL	Help	00:12:30	2022-06-15
	US0002	SR0010	2	NULL	What?	00:13:30	2022-06-15
	US0002	SR0010	3	5	I love her	00:14:30	2022-06-15
	US0002	SR0010	4	NULL	I hate her	00:15:30	2022-06-15
	US0002	SR0010	5	NULL	5555	00:16:30	2022-06-15
	US0002	SR0010	6	5	I love it	00:17:30	2022-06-15
	US0002	SR0009	1	NULL	NULL	00:18:30	2022-06-15
	US0002	SR0009	2	NULL	NULL	00:19:30	2022-06-15
	US0002	SR0009	3	NULL	NULL	00:20:30	2022-06-15
	US0002	SR0009	4	NULL	NULL	00:21:30	2022-06-15
	US0002	SR0009	5	NULL	NULL	00:22:30	2022-06-15
	US0002	SR0009	6	5	I love it	00:23:30	2022-06-15
	US0003	SR0022	9	NULL	NULL	00:04:00	2022-01-22
	US0003	SR0017	1	5	5555	00:04:00	2022-03-04
	US0003	SR0024	5	NULL	NULL	00:04:00	2022-12-09
	US0003	SR0007	11	NULL	What is it?	00:04:00	2022-07-25
	US0004	SR0001	1	NULL	NULL	00:01:30	2022-04-20
	US0004	SR0001	2	NULL	NULL	00:03:30	2022-05-01
	US0005	SR0001	1	5	Why	00:08:08	2022-04-20
	US0005	SR0001	4	5	NULL	00:09:30	2022-04-20
	US0005	SR0001	9	5	NULL	00:10:15	2022-04-20
	US0005	SR0001	10	5	NULL	00:00:03	2022-04-21
	US0005	SR0001	22	5	i love him	00:22:00	2022-04-21
	US0005	SR0001	49	5	NULL	00:22:50	2022-04-21
	US0006	SR0025	4	NULL	NULL	00:10:30	2022-04-18
	US0006	SR0025	5	5	Cool	00:11:30	2022-04-18
	US0006	SR0011	1	NULL	NULL	00:00:08	2022-06-20
	US0006	SR0011	2	NULL	NULL	00:01:32	2022-06-20
	US0006	SR0011	3	4	Sweet	00:02:30	2022-06-21
	US0007	SR0022	3	5	Very Good	00:01:00	2022-11-15
	US0008	SR0003	1	4	Good	00:09:00	2022-06-04
	US0008	SR0003	24	4	I love him	00:09:00	2022-06-04
	US0009	SR0017	1	5	Good	00:15:30	2022-08-04
	US0009	SR0017	20	4	It OK	00:16:00	2022-08-04
	US0009	SR0017	34	3	NULL	00:16:20	2022-08-04
	US0010	SR0018	38	5	Sweet	00:14:00	2022-01-13

3- series

INSERT INTO series (Series_Id, Series_Name, Series_TypeId, Num_Episodes, Release_Year, Country, Rating) VALUES

('SR0001', 'The Untamed', 'ST0006', '50', '2019', 'China', '5'),

('SR0002', 'Falling Into Your Smile', 'ST0005', '31', '2022', 'China', '4'),

('SR0003', 'The Romance of Tiger and Rose', 'ST0005', '24', '2020', 'China', '5'),

('SR0004', 'Together The Series', 'ST0005', '3', '2020', 'Thai', '4'),

('SR0005', 'Love Mechanics', 'ST0005', '25', '2020', 'Thai', '3'),

('SR0006', 'KinnPorsche The Series', 'ST0006', '17', '2020', 'Thai', '4'),

('SR0007', 'Descendants of the Sun', 'ST0005', '16', '2016', 'Korea', '5'),

('SR0008', 'Big Mouth', 'ST0004', '16', '2022', 'Korea', '3'),

('SR0009', 'Kingdom 2', 'ST0004', '6', '2022', 'Korea', '5'),

('SR0010', 'Kingdom 1', 'ST0004', '6', '2019', 'Korea', '4'),

('SR0011', 'My Country', 'ST0002', '16', '2019', 'Korea', '3'),

('SR0012', 'The Eternal Monarch', 'ST0002', '16', '2020', 'Korea', '4'),

('SR0013', 'The Tale of Nokdu', 'ST0002', '32', '2019', 'Korea', '5'),

('SR0014', 'Vagabond', 'ST0006', '32', '2019', 'Korea', '5'),

('SR0015', 'Sweet Tooth', 'ST0003', '8', '2021', 'England', '5'),

('SR0016', 'The Queen's Gambit', 'ST0003', '7', '2020', 'England', '4'),

('SR0017', 'Stranger Things', 'ST0003', '34', '2016', 'England', '3'),

('SR0018', 'Game of Thrones', 'ST0003', '78', '2011', 'England', '5'),

4- series_type

INSERT INTO series_type (Series_TypeId, TypeName) VALUES

('ST0001', 'Drama'),

('ST0002', 'Historical'),

('ST0003', 'Fantasy'),

('ST0004', 'Suspense'),

('ST0005', 'Romantic'),

('ST0006', 'Action');

	Series_TypeId	TypeName
▶	ST0001	Drama
	ST0002	Historical
	ST0003	Fantasy
	ST0004	Suspense
	ST0005	Romantic
	ST0006	Action
•	NULL	NULL

5- user_has_package

INSERT INTO user_has_package (User_PayID, User_Id, Package_Id, Payment_Id, Package_Startdate, Package_Enddate, Payment_Time) VALUES

('UP0001', 'US0002', 'PK0002', 'PM0002', '2022-02-10', '2022-03-12', '00:22:40'),

('UP0002', 'US0003', 'PK0001', 'PM0001', '2022-05-15', '2022-06-16', '00:12:35'),

('UP0003', 'US0003', 'PK0002', 'PM0001', '2022-06-15', '2022-07-15', '00:00:18'),

('UP0004', 'US0002', 'PK0001', 'PM0003', '2022-09-01', '2022-10-01', '00:07:30'),

('UP0005', 'US0001', 'PK0002', 'PM0002', '2022-11-21', '2022-12-21', '00:10:30'),
 ('UP0006', 'US0004', 'PK0001', 'PM0002', '2022-08-23', '2022-09-22', '00:13:30'),
 ('UP0007', 'US0005', 'PK0002', 'PM0002', '2022-04-24', '2022-05-24', '00:00:30'),
 ('UP0008', 'US0006', 'PK0001', 'PM0001', '2022-05-12', '2022-06-13', '00:23:30'),
 ('UP0009', 'US0007', 'PK0002', 'PM0002', '2022-03-18', '2022-04-19', '00:21:30'),
 ('UP0010', 'US0008', 'PK0001', 'PM0003', '2022-01-31', '2022-03-02', '00:11:01');

	User_PayID	User_Id	Package_Id	Payment_Id	Package_Startdate	Package_Enddate	Payment_Time
▶	UP0001	US0002	PK0002	PM0002	2022-02-10	2022-03-12	00:22:40
	UP0002	US0003	PK0001	PM0001	2022-05-15	2022-06-16	00:12:35
	UP0003	US0003	PK0002	PM0001	2022-06-15	2022-07-15	00:00:18
	UP0004	US0002	PK0001	PM0003	2022-09-01	2022-10-01	00:07:30
	UP0005	US0001	PK0002	PM0002	2022-11-21	2022-12-21	00:10:30
	UP0006	US0004	PK0001	PM0002	2022-08-23	2022-09-22	00:13:30
	UP0007	US0005	PK0002	PM0002	2022-04-24	2022-05-24	00:00:30
	UP0008	US0006	PK0001	PM0001	2022-05-12	2022-06-13	00:23:30
	UP0009	US0007	PK0002	PM0002	2022-03-18	2022-04-19	00:21:30
	UP0010	US0008	PK0001	PM0003	2022-01-31	2022-03-02	00:11:01
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6- payment_method

INSERT INTO payment_method (Payment_Id, Payment_Name) VALUES

('PM0001', 'trueMoney'),
 ('PM0002', 'Prompt Pay'),
 ('PM0003', 'Bank');

	Payment_Id	Payment_Name
▶	PM0001	trueMoney
	PM0002	Prompt Pay
	PM0003	Bank
•	NULL	NULL

7- package

```
INSERT INTO package (Package_Id, Package_Name, Price_month, Benefit) VALUES  
  
('PK0001', 'VIP', '100', '1089P'),  
  
('PK0002', 'VIP Premium', '500', '4K, Save VDO ');
```

	Package_Id	Package_Name	Price_month	Benefit
▶	PK0001	VIP	100	1089P
	PK0002	VIP Premium	500	4K, Save VDO
•	NULL	NULL	NULL	NULL