FEB UNPAD
LEADING AND INSPIRING

FAKULTAS EKONOMI DAN BISNIS UNIVERSITAS PADJADJARAN
PROGRAM STUDI BISNIS DIGITAL

UJIAN AKHIR SEMESTER

MATA KULIAH	: B10E-3103 Sistem Basis Data
HARI / TANGGAL	: Senin / 6 Juni 2022
SEMESTER	: Genap 2021
DOSEN	: Andria ARISAL, Ana HERYANA
WAKTU	: Senin, 6 Juni 2022 10:30 - Sabtu 11 Juni 2022 23:59
TEMPAT	: rumah
SIFAT UJIAN	: Take home, open book, dan online submission

Nama	Akinla Diko Omari
No. KTM	120510210036

- Baca instruksi berikut sebelum mulai mengerjakan jawaban.
- Ujian ini terdiri dari empat bagian, dengan informasi nilai maksimum di setiap soal.
- Ujian dilakukan secara mandiri selama lima (5) hari dimulai dari waktu jadwal ujian pada hari Senin, tanggal 6 Juni 2022, sampai tengah malam pada hari Sabtu, tanggal 11 Juni 2022, pukul 23.59.
- Jawaban dituliskan dan disimpan dalam bentuk satu file PDF yang berisi jawaban dan satu file SQL yang berisi kode yang bisa langsung dimasukkan dan dijalankan di basis data relasional yang digunakan. Tuliskan sistem basis data relasional yang digunakan beserta versinya di awal file SQL yang dikirimkan, sebagai komentar (diawali dengan dua tanda -)
- Contoh:

```
-- MySQL 8.0.23
```

- Untuk menyelesaikan tugas ini, buat sebuah skema basis data baru bernama UAS_[NPM] di H2 database atau di MySQL atau di PostgreSQL.
- Contoh:

```
CREATE SCHEMA UAS_123456789012;
```

- Setiap file harus bernama UAS_[NPM].pdf dan UAS_[NPM].sql. Contoh: UAS_12345678.pdf dan UAS_12345678.sql
- Hanya gunakan google classroom untuk mengirimkan jawaban. Penggunaan metode pengiriman jawaban lain tidak diterima.

- Sistem akan tetap menerima jawaban yang lewat dari batas waktu yang ditentukan, dan mencatat waktu keterlambatan. Setiap menit keterlambatan akan mengurangi nliai. Untuk itu sediakan waktu yang cukup untuk menghindari hal-hal yang tidak diinginkan. Mengupload jawaban lebih cepat, akan lebih baik.
- Jika terdapat dua jawaban yang sama dan mengindikasikan terjadinya kerja sama, maka yang mengunggah (upload) lebih dulu yang mendapatkan nilai, sedangkan yang mengunggah belakangan akan dinilai 0 (NOL). Serta akan dilaporkan ke Ketua Program Studi.
- Contoh beberapa baris pada file SQL yang dikumpulkan:

```
-- UAS B10E-3103 2022
-- Holly Wood, 123456789012
-- MySQL 8.0.23

CREATE SCHEMA 123456789012;

-- 1. Model konseptual. Diagram di file PDF
-- 2. Model logika.
-- MEMBER(id: string, name: string, member_type_id ->
-- MEMBER_TYPE.id, ...)
-- 3.
CREATE …
```

Jangan menulis dibawah garis ini

1	2	3	4	Total	Nilai

0. Deskripsi Sistem

Sebuah perusahaan rintisan (FamilyVideo.com) yang bergerak dibidang penyewaan film menyediakan layanan yang memungkinkan anggota untuk menonton film yang diinginkannya secara daring.

Perusahaan menyimpan data anggota seperti nomor anggota, nama, alamat, kota atau kabupaten, provinsi, nomor telepon, email, tanggal mulai keanggotaan, tanggal akhir keanggotaan, metode pembayaran, dan lainnya. Setiap anggota memiliki tingkat keanggotaannya yaitu “Bronze” dengan iuran keanggotaan sebesar Rp. 50.000 per bulan belum termasuk pajak 11% yang dapat menonton sebanyak 1 film dalam satu waktu; “Silver” dengan iuran keanggotaan sebesar 75.000 per bulan belum termasuk pajak, yang dapat menonton film sebanyak 2 film dalam satu waktu; dan “Gold” dengan iuran keanggotaan sebesar 90.000 per bulan belum termasuk pajak, yang dapat menonton film sebanyak 4 film dalam satu waktu. Sistem mencatat informasi cara pembayaran, tagihan, dan pelunasan untuk setiap anggotanya.

Perusahaan memiliki koleksi film yang sangat banyak. Setiap film memiliki informasi kode film, jenis film, judul film, judul asli dari film, klasifikasi usia film, tahun rilis film, panjang durasi film. Klasifikasi usia film dan panjang durasi film bisa belum diketahui. Setiap film bisa memiliki satu, atau lebih genre, atau bisa juga memiliki genre yang belum diketahui.

Jika diketahui, sistem juga mencatat bahwa setiap film dapat memiliki sejumlah insan perfilman sesuai dengan perannya misalkan sebagai aktor/aktris serta beberapa kru film sesuai dengan fungsinya seperti (director, writer, producer, editor, music_department, animator, soundtrack, dll).

Setiap orang yang berhubungan dengan film memiliki informasi nama, tahun lahir, tahun meninggal, kewarganegaraan.

Setiap anggota dapat menonton banyak film yang tersedia. Untuk setiap film yang ditonton oleh seorang anggota, sistem akan mencatat waktu awal menonton, waktu yang dilewatkan dalam menonton film tersebut, dan jika anggota tersebut mau, dia juga memberikan rating berupa nilai 1 sampai 5, dan teks review atas film yang sudah ditontonnya.

Jenis film adalah:

- Short: film pendek
- Movie: film panjang
- TvEpisode: satu episode dari serial TV
- TvSeries: satu serial TV
- TvShort: film pendek untuk TV
- TvMovie: film panjang untuk TV
- VideoGame: film dari video game

Klasifikasi usia film adalah:

- G: Khalayak umum (*General Audiences*)
  Diperuntukkan bagi semua umur. Tiada yang patut dikhawatirkan orang tua bilamana anak-anak menonton film ini.
- PG - Pendampingan orang tua disarankan (*Parental Guidance Suggested*)
  Sebagian isi film mungkin tidak sesuai untuk anak-anak. Orang tua diminta memberikan "pendampingan orang tua" kepada anak-anak mereka. Isi film bisa jadi berisi beberapa hal yang mungkin tidak diharapkan orang tua bagi anak-anak mereka.
- PG-13: Peringatan keras bagi orang tua (*Parents Strongly Cautioned*)
  Sebagian isi film mungkin tidak sesuai untuk anak-anak berusia di bawah 13 tahun. Orang tua diminta berhati-hati. Beberapa hal yang terkandung dalam sebuah film mungkin tidak sesuai bagi remaja.
- R: Terbatas (*Restricted*)
  Orang berusia di bawah 17 tahun harus ditemani orang tua atau wali. Mengandung beberapa hal yang bersifat dewasa. Orang tua diminta mempelajari lebih lanjut perihal film ini sebelum mengajak anak-anak menonton film bersama mereka.
- NC-17: Hanya untuk dewasa (*No One 17 and Under Admitted*)
  Tidak diperuntukkan bagi orang berusia 17 tahun ke bawah. Benar-benar untuk dewasa. Anak-anak tidak diperkenankan untuk menonton film ini.

Genre film meliputi:

- Action
- Adventure
- Animation
- Biography
- Comedy
- Crime
- Documentary
- Drama
- Family
- Fantasy
- HIstory
- Horror
- Mystery
- Music
- Romance
- Sci-Fi
- Sport
- Thriller
- War
- Western

1. Model Konseptual [15]

[5] Uraikanlah dan tambahkan informasi dan asumsi jika diperlukan dari deskripsi sistem yang dijelaskan pada bagian 0.

Jawaban:

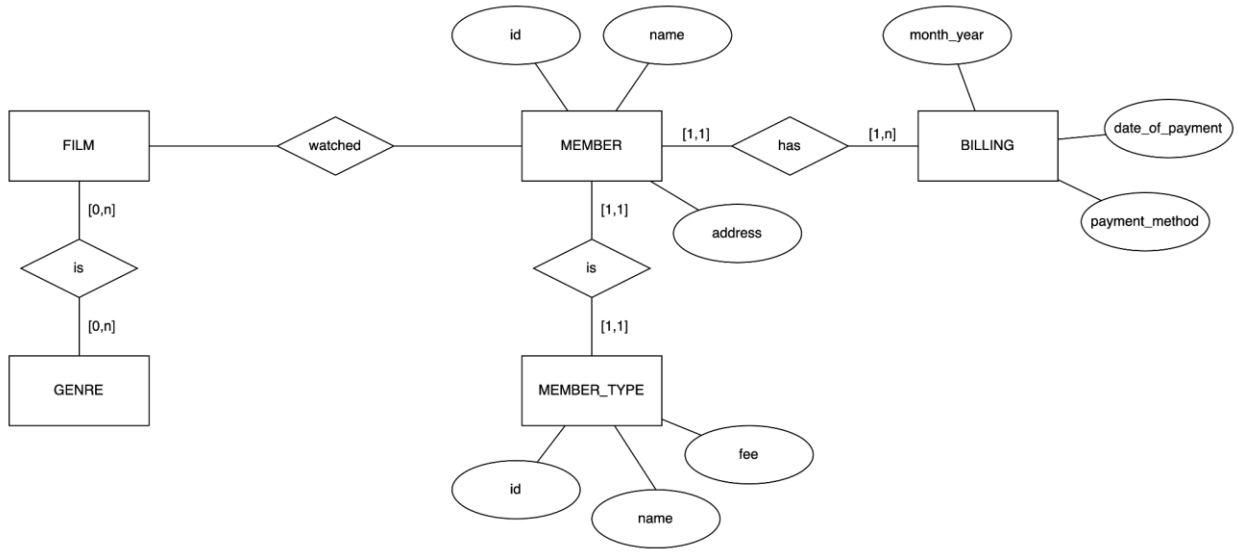
Sistem yang akan diulas pada UAS ini adalah start-up FamilyVideo.com yang bergerak di bidang penyewaan film daring.

Entitas pertama yang dapat diidentifikasi adalah anggota. Entitas ini memiliki atribut nomor anggota, nama, alamat, kota atau kabupaten, provinsi, nomor telepon, email, tanggal mulai keanggotaan, tanggal akhir keanggotaan, metode pembayaran, cara pembayaran, tagihan, dan tanggal pelunasan. Entitas anggota memiliki entitas lemah yang diturunkan yaitu tingkat keanggotaan. Tingkat keanggotaan memiliki atribut tipe keanggotaan, iuran, pajak, dan jumlah film dalam satu waktu

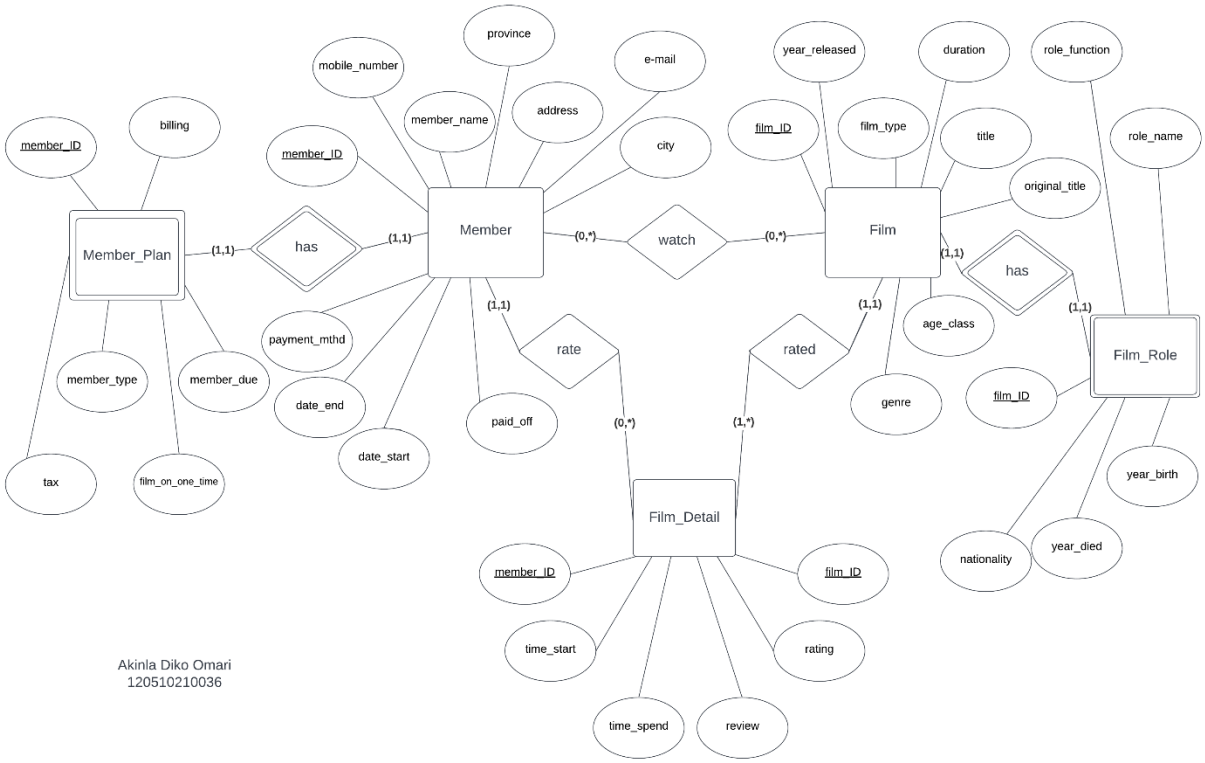
Entitas selanjutnya adalah film. Film mencatat atribut kode film, jenis film, judul film, judul asli dari film, klasifikasi usia film, tahun rilis film, panjang durasi film, dan genre film. Entitas ini mempunyai entitas lemah yaitu insan perfilman. Entitas insan perfilman mencatat kode film, fungsi peran, nama, tahun lahir, tahun meninggal, dan kewarganegaraan.

Entitas selanjutnya adalah detail. Entitas ini menggunakan nomor anggota dan kode film sebagai atribut unik. Detail mencatat informasi waktu awal menonton, waktu yang dilewatkan untuk menonton, rating film, dan review film.

[10] Buatlah model konseptual dalam bentuk Entity Relationship Diagram dengan menguraikan semua entitas, atribut, dan relasi yang ada serta menunjukkan kunci, dan kardinalitas dari relasi dalam bentuk nilai minimum dan maksimum. Lengkapi bagian ER-Diagram yang sudah tersedia atau buat ER-Diagram yang baru dan lebih lengkap.



Jawaban:



2. Model Logika [15]

[15] Buatlah model logika dalam bentuk notasi relational model dari ER-diagram yang sudah dibuat. Pastikan semua tabel dengan semua atributnya sudah dibuat dan semua relasi yang diuraikan pada ER-diagram sudah diakomodasi dalam tabel-tabel yang dibuat. Tuliskan juga tipe data untuk setiap atribut dari tabel dan tunjukkan dengan notasi yang sesuai jika diperlukan foreign key bahwa atribut mengacu kepada atribut di tabel lain. Pastikan rancangan sudah memenuhi kaidah Normalisasi ketiga (3NF).

Contoh:
MEMBER(id: string, name: string, member_type_id -> MEMBER_TYPE.id, ...)
MEMBERTYPE(id: string, name: string, fee: integer)

Jawaban:

UNF:
Member(member_ID: integer, member_name: varchar, address: varchar, city: varchar, province: varchar, email: varchar, mobile_number: integer, payment_mthd: varchar, date_start: date, date_end: date, paid_off: date)

Member_Plan(member_ID: integer -> Member.member_ID, member_type: varchar, member_due: integer, tax: integer, film_on_one_time: tinyint, billing: integer)

Film(film_ID: integer, title: varchar, original_title: varchar, film_type: varchar, duration: integer, year_released: year, age_class: varchar, genre: varchar)

Film_Role(film_ID: integer -> Film.film_ID, role_name: varchar, role_function: varchar, nationality: varchar, year_birth: year, year_died: year)

Film_Detail(member_ID: integer -> Member.member_ID, film_ID: integer -> Film.film_ID, time_start: datetime, time_spend: int, review: text, rating: tinyint)

1NF:
Member(member_ID: integer, member_first_name: varchar, member_last_name: varchar, address: varchar, city: varchar, province: varchar, email: varchar, mobile_number: integer, payment_mthd: varchar, date_start: date, date_end: date, paid_off: date)

Member_Plan(member type: varchar, member ID: integer -> Member.member_ID, member_due: integer, tax: integer, film_on_one_time: tinyint, billing: integer)

Film(film ID: integer, title: varchar, original_title: varchar, film_type: varchar, duration: integer, year_released: year, age_class: varchar, genre: varchar)

Film_Role(film ID: integer -> Film.film_ID, role first name: varchar, role last name: varchar, role function: varchar, nationality: varchar, year_birth: year, year_died: year)

Film_Detail(member ID: integer -> Member.member_ID, film ID: integer -> Film.film_ID, time_start: datetime, time_spend: int, review: text, rating: tinyint)

2NF:

Member_Bio(member ID: integer, member_first_name: varchar, member_last_name: varchar, address: varchar, city: varchar, province: varchar, email: varchar, mobile_number: integer)

Member_Start(paid off: date, date_start: date, date_end: date)

Member_Payment(member ID: integer -> Member_Bio.member_ID, paid off: date -> Member_Start.paid_off, payment_mthd: varchar)

Member_Level(member type: varchar, member_due: integer, tax: integer, film_on_one_time: tinyint, billing: integer)

Member_Plan(member ID: integer -> Member_Bio.member_ID, paid off: date -> Member_Start.paid_off, member_type: varchar -> Member_Level.member_type)

Film(film ID: integer, title: varchar, original_title: varchar, film_type: varchar, duration: integer, year_released: year, age_class: varchar, genre: varchar)

Role_Bio(role_first_name: varchar, role_last_name: varchar, role_function: varchar, nationality: varchar, year_birth: year, year_died: year)

Film_Role(film ID: integer -> Film.film_ID, role_first_name: varchar -> Role_Bio.role_first_name, role_last_name: varchar -> Role_Bio.role_last_name, role_function: varchar -> Role_Bio.role_function)

Film_Detail(member_ID: integer -> Member_Bio.member_ID, film_ID: integer -> Film.film_ID, time_start: datetime, time_spend: int, review: text, rating: tinyint)

3NF:

Member_Bio(email: varchar, member_first_name: varchar, member_last_name: varchar, address: varchar, city: varchar, province: varchar, mobile_number: varchar)

Member_List(member_ID: integer, email: varchar -> Member_Bio.email)

Member_Start(paid_off: date, date_start: date, date_end: date)

Member_Payment(member_ID: integer -> Member_List.member_ID, paid_off: date -> Member_Start.paid_off, payment_mthd: varchar)

Level_Price(billing: integer, member_due: integer, tax: integer)

Member_Level(member_type: varchar, film_on_one_time: tinyint, billing: integer -> Level_Price.billing)

Member_Plan(member_ID: integer -> Member_List.member_ID, paid_off: date -> Member_Start.paid_off, member_type: varchar -> Member_Level.member_type)

Film_Bio(title: varchar, original_title: varchar, film_type: varchar, duration: integer, year_released: year, age_class: varchar, genre: varchar)

Film_List(film_ID: integer, title: varchar -> Film_Bio.title)

Role_Bio(role_first_name: varchar, role_last_name: varchar, role_function: varchar, nationality: varchar, year_birth: year, year_died: year)

Film_Role(film_ID: integer -> Film.film_ID, role_first_name: varchar -> Role_Bio.role_first_name, role_last_name: varchar -> Role_Bio.role_last_name, role_function: varchar -> Role_Bio.role_function)

Film_Detail(member_ID: integer -> Member_List.member_ID, film_ID: integer -> Film_List.film_ID, time_start: datetime, time_spend: int, review: text, rating: tinyint)

3. Model Fisik [25]

[15] Buatlah perintah-perintah SQL yang akan membuat semua tabel yang diuraikan pada model logika pada bagian ke-2.

Contoh:

```
-- UAS B10E-3103 2022
-- Holly Wood, 123456789012
-- MySQL 8.0.23

CREATE SCHEMA UAS_123456789012;
USE UAS_123456789012

CREATE TABLE MEMBER (
      id VARCHAR(10) NOT NULL PRIMARY KEY,
      fee INTEGER NOT NULL
);

CREATE TABLE MEMBER (
      id VARCHAR(20) NOT NULL,
      name VARCHAR(50) NOT NULL,
      member_type_id VARCHAR(10),
      ...
      PRIMARY KEY (id)
      CONSTRAINT FK_membertype FOREIGN KEY (member_type_id)
          REFERENCES MEMBERTYPE(id)

);
```

Jawaban:

```
drop schema if exists UAS_120510210036;
create schema UAS_120510210036;
use UAS_120510210036;

create table Member_Bio (
      email varchar(50) not null,
      member_first_name varchar(20) not null,
      member_last_name varchar(20) not null,
      address varchar(50) not null,
      city varchar(50) not null,
      province varchar(50) not null,
      mobile_number varchar(20) not null,
      primary key (email)
);

create table Member_List (
```

```
	member_ID int not null auto_increment,
	email varchar(50) not null,
	primary key (member_ID),
	foreign key (email) references Member_Bio(email)
);
alter table Member_List auto_increment=36001;

create table Member_Start (
	paid_off date not null,
	date_start date not null,
	date_end date not null,
	primary key (paid_off)
);

create table Member_Payment (
	member_ID int not null auto_increment,
	paid_off date not null,
	payment_method varchar(50) not null,
	primary key (member_ID, paid_off),
	foreign key (member_ID) references Member_List(member_ID),
	foreign key (paid_off) references Member_Start(paid_off)
);
alter table Member_Payment auto_increment=36001;

create table Level_Price (
	billing int not null,
	member_due int not null,
	tax int not null,
	primary key (billing)
);

create table Member_Level (
	member_type varchar(10) not null,
	film_on_one_time tinyint not null,
	billing int not null,
	primary key (member_type),
	foreign key (billing) references Level_Price(billing)
);

create table Member_Plan (
	member_ID int not null auto_increment,
	paid_off date not null,
	member_type varchar(10) not null,
	primary key (member_ID, paid_off),
	foreign key (member_ID) references Member_List(member_ID),
	foreign key (paid_off) references Member_Start(paid_off),
	foreign key (member_type) references Member_Level(member_type)
);
alter table Member_Plan auto_increment=36001;

create table Film_Bio (
	title varchar(200) not null,
	original_title varchar(300) not null,
	film_type varchar(20) not null,
	duration int,
```

```
	year_released year not null,
	age_class varchar(10),
	genre varchar(20),
	primary key (title)
);

create table Film_List (
	film_ID int not null auto_increment,
	title varchar(200) not null,
	primary key (film_ID),
	foreign key (title) references Film_Bio(title)
);
alter table Film_List auto_increment=63101;

create table Role_Bio (
	role_first_name varchar(30) not null,
	role_last_name varchar(30) not null,
	role_function varchar(20) not null,
	nationality varchar(20) not null,
	year_birth year not null,
	year_died year,
	primary key (role_first_name, role_last_name, role_function)
);

create table Film_Role (
	film_ID int not null,
	role_first_name varchar(30) not null,
	role_last_name varchar(30) not null,
	role_function varchar(20) not null,
	primary key (film_ID, role_first_name, role_last_name, role_function),
	foreign key (film_ID) references Film_List(film_ID),
	foreign key (role_first_name, role_last_name, role_function) references
Role_Bio(role_first_name, role_last_name, role_function)
);

create table Film_Detail (
	member_ID int not null auto_increment,
	film_ID int not null,
	time_start datetime not null,
	time_spend int,
	review text,
	rating tinyint,
	primary key (member_ID, film_ID),
	foreign key (member_ID) references Member_List(member_ID),
	foreign key (film_ID) references Film_List(film_ID)
);
alter table Film_Detail auto_increment=36001;
```

[5] Isilah setiap tabel yang sudah dibuat dengan paling sedikit 20 contoh data yang sesuai. Gunakan kreatifitas masing-masing.

Jawaban:

```
insert into Member_Bio (email, member_first_name, member_last_name, address, city,
province, mobile_number) values
('dikoooo@gmail.com', 'diko', 'omar', 'condet', 'jakarta timur', 'jakarta',
'6281200010001'),
('cr7@gmail.com', 'cristiano', 'ronaldo', 'old trafford', 'manchester', 'greater
manchester', '6281200020002'),
('leo10messi@gmail.com', 'lionel', 'messi', 'parc des princes', 'paris', 'ile-de-
france', '6281200030003'),
('lilb0ss@gmail.com', 'kylian', 'mbappe', 'parc des princes', 'paris', 'ile-de-
france', '6281200040004'),
('wakhaji@gmail.com', 'karim', 'benzema', 'santiago bernabeu', 'madrid', 'madrid',
'6281200050005'),
('lewangoalski@gmail.com', 'robert', 'lewandowski', 'allianz arena', 'munchen',
'bayern', '6281200060006'),
('tidakbenar@gmail.com', 'mohamed', 'salah', 'anfield', 'liverpool', 'merseyside',
'6281200070007'),
('kingibra@gmail.com', 'zlatan', 'ibrahimovic', 'san siro', 'milan', 'lombardy',
'6281200080008'),
('meguayer@gmail.com', 'harry', 'maguire', 'old trafford', 'manchester', 'greater
manchester', '6281200090009'),
('notrophy@gmail.com', 'harry', 'kane', 'tottenham hotspur stadium', 'north london',
'london', '6281200100010'),
('touchmaster@gmail.com', 'romelu', 'lukaku', 'stamford bridge', 'fulham', 'london',
'6281200110011'),
('realgoat@gmail.com', 'martin', 'braithwaite', 'camp nou', 'barcelona', 'catalonia',
'6281200120012'),
('baggoat@gmail.com', 'elkan', 'baggot', 'portman road', 'ipswich', 'suffolk',
'6281200130013'),
('legendarykeeper@gmail.com', 'loris', 'karius', 'anfield', 'liverpool',
'merseyside', '6281200140014'),
('thecarry@gmail.com', 'david', 'de gea', 'old trafford', 'manchester', 'greater
manchester', '6281200150015'),
('playmaker@gmail.com', 'kevin', 'de bruyne', 'etihad stadium', 'manchester',
'greater manchester', '6281200160016'),
('gajibuta@gmail.com', 'gareth', 'bale', 'santiago bernabeu', 'madrid', 'madrid',
'6281200170017'),
('vinisicepat@gmail.com', 'vinicius', 'junior', 'santiago bernabeu', 'madrid',
'madrid', '6281200180018'),
('vampire@gmail.com', 'luis', 'suarez', 'wanda metropolitano', 'madrid', 'madrid',
'6281200190019'),
('thelord@gmail.com', 'nicklas', 'bendtner', 'emirates stadium', 'holloway',
'london', '6281200200020');

insert into Member_List (email) values
('dikoooo@gmail.com'),
('cr7@gmail.com'),
('leo10messi@gmail.com'),
('lilb0ss@gmail.com'),
('wakhaji@gmail.com'),
```

```
('lewangoalski@gmail.com'),
('tidakbenar@gmail.com'),
('kingibra@gmail.com'),
('meguayer@gmail.com'),
('notrophy@gmail.com'),
('touchmaster@gmail.com'),
('realgoat@gmail.com'),
('baggoat@gmail.com'),
('legendarykeeper@gmail.com'),
('thecarry@gmail.com'),
('playmaker@gmail.com'),
('gajibuta@gmail.com'),
('vinisicepat@gmail.com'),
('vampire@gmail.com'),
('thelord@gmail.com');

insert into Member_Start (paid_off, date_start, date_end) values
('2022-06-01', '2022-06-01', '2022-07-01'),
('2022-06-02', '2022-06-02', '2022-07-02'),
('2022-06-03', '2022-06-03', '2022-07-03'),
('2022-06-04', '2022-06-04', '2022-07-04'),
('2022-06-05', '2022-06-05', '2022-07-05'),
('2022-06-06', '2022-06-06', '2022-07-06'),
('2022-06-07', '2022-06-07', '2022-07-07'),
('2022-06-08', '2022-06-08', '2022-07-08'),
('2022-06-09', '2022-06-09', '2022-07-09'),
('2022-06-10', '2022-06-10', '2022-07-10'),
('2022-06-11', '2022-06-11', '2022-07-11'),
('2022-06-12', '2022-06-12', '2022-07-12'),
('2022-06-13', '2022-06-13', '2022-07-13'),
('2022-06-14', '2022-06-14', '2022-07-14'),
('2022-06-15', '2022-06-15', '2022-07-15'),
('2022-06-16', '2022-06-16', '2022-07-16'),
('2022-06-17', '2022-06-17', '2022-07-17'),
('2022-06-18', '2022-06-18', '2022-07-18'),
('2022-06-19', '2022-06-19', '2022-07-19'),
('2022-06-20', '2022-06-20', '2022-07-20');

insert into Member_Payment (paid_off, payment_method) values
('2022-06-03', 'ovo'),
('2022-06-15', 'gopay'),
('2022-06-20', 'shopeepay'),
('2022-06-04', 'familyvideos wallet'),
('2022-06-09', 'gopay'),
('2022-06-15', 'familyvideos wallet'),
('2022-06-13', 'indomaret'),
('2022-06-01', 'gopay'),
('2022-06-05', 'debit card'),
('2022-06-08', 'credit card'),
('2022-06-08', 'gopay'),
('2022-06-19', 'ovo'),
('2022-06-18', 'credit card'),
('2022-06-16', 'debit card'),
('2022-06-19', 'debit card'),
('2022-06-20', 'alfamart'),
```

```
('2022-06-06', 'shopeepay'),
('2022-06-11', 'credit card'),
('2022-06-02', 'ovo'),
('2022-06-10', 'gopay');

insert into Level_Price (billing, member_due, tax) values
(55500, 50000, 5500),
(83250, 75000, 8250),
(99900, 90000, 9900);

insert into Member_Level (member_type, film_on_one_time, billing) values
('bronze', 1, 55500),
('silver', 2, 83250),
('gold', 4, 99900);

insert into Member_Plan (paid_off, member_type) values
('2022-06-03', 'silver'),
('2022-06-15', 'gold'),
('2022-06-20', 'gold'),
('2022-06-04', 'bronze'),
('2022-06-09', 'silver'),
('2022-06-15', 'silver'),
('2022-06-13', 'gold'),
('2022-06-01', 'gold'),
('2022-06-05', 'gold'),
('2022-06-08', 'silver'),
('2022-06-08', 'bronze'),
('2022-06-19', 'silver'),
('2022-06-18', 'gold'),
('2022-06-16', 'bronze'),
('2022-06-19', 'bronze'),
('2022-06-20', 'bronze'),
('2022-06-06', 'silver'),
('2022-06-11', 'gold'),
('2022-06-02', 'gold'),
('2022-06-10', 'silver');

insert into Film_Bio(title, original_title, film_type, duration, year_released,
age_class, genre) values
('The Shawshank Redemption', 'Rita Hayworth and the Shawshank Redemption', 'Movie',
142, '1994', 'R', 'Drama'),
('Inception', 'Inception', 'Movie', 148, '2010', 'PG-13', 'Action'),
('Interstellar', 'Interstellar', 'Movie', 169, '2014', 'PG-13', 'Sci-Fi'),
('The Matrix', 'The Matrix', 'Movie', 136, '1999', 'R', 'Action'),
('The Wolf of Wall Street', 'The Wolf of Wall Street', 'Movie', 180, '2013', 'R',
'Biography'),
('The Lion King', 'The Lion King', 'Movie', 88, '1994', 'G', 'Animation'),
('Stranger Things', 'Stranger Things', 'TVSeries', 51, '2016', 'TV-14', 'Drama'),
('Game of Thrones S1', 'Game of Thrones', 'TV Series', 57, '2011', 'TV-MA', 'Drama'),
('Live Free or Die', 'Breaking Bad', 'TV Episodes', 43, '2012', 'TV-14', 'Crime'),
('Dead Freight', 'Breaking Bad', 'TV Episodes', 48, '2012', 'TV-14', 'Crime'),
('The Godfather', 'The Godfather', 'Movie', 175, '1972', 'R', 'Crime'),
('Coco', 'Coco', 'Movie', 105, '2017', 'PG', 'Animation'),
('Your Name.', 'Kimi no na wa', 'TV Series', 106, '2016', 'TV-PG', 'Animation'),
('3 Idiots', '3 Idiots', 'Movie', 170, '2009', 'PG-13', 'Comedy'),
```

```
('The Dark Knight', 'The Dark Knight', 'Movie', 152, '2008', 'PG-13', 'Action'),
('Gladiator', 'Gladiator', 'Movie', 155, '2000', 'R', 'Action'),
('Joker', 'Joker', 'Movie', 122, '2019', 'R', 'Thriller'),
('Return of the Jedi', 'Star Wars: Episode VI', 'Movie', 131, '1983', 'PG',
'Fantasy'),
('Whiplash', 'Whiplash', 'Movie', 166, '2014', 'R', 'Music'),
('Jaws', 'Jaws', 'Movie', 124, '1975', 'PG', 'Adventure');

insert into Film_List (title) values
('The Shawshank Redemption'),
('Inception'),
('Interstellar'),
('The Matrix'),
('The Wolf of Wall Street'),
('The Lion King'),
('Stranger Things'),
('Game of Thrones S1'),
('Live Free or Die'),
('Dead Freight'),
('The Godfather'),
('Coco'),
('Your Name.'),
('3 Idiots'),
('The Dark Knight'),
('Gladiator'),
('Joker'),
('Return of the Jedi'),
('Whiplash'),
('Jaws');

insert into Role_Bio (role_first_name, role_last_name, role_function, nationality,
year_birth) values
('Christopher', 'Nolan', 'director', 'England', '1970'),
('Christopher', 'Nolan', 'writer', 'England', '1970'),
('Frank', 'Darabont', 'director', 'France', '1959'),
('Stephen', 'King', 'writer', 'USA', '1947'),
('Jonathan', 'Nolan', 'writer', 'England', '1976'),
('Darla', 'Anderson', 'producer', 'USA', '1968' ),
('Lee', 'Unkrich', 'editor', 'USA', '1967'),
('Rajkumar', 'Hirani', 'director', 'India', '1962'),
('Rajkumar', 'Hirani', 'writer', 'India', '1962'),
('Hildur', 'Guðnadóttir', 'music', 'Iceland', '1982'),
('Makoto', 'Shinkai', 'director', 'Japan', '1973'),
('Makoto', 'Shinkai', 'writer', 'Japan', '1973'),
('Makoto', 'Shinkai', 'art', 'Japan', '1973'),
('Laurie', 'MacDonald', 'producer', 'USA', '1953'),
('Hans', 'Zimmer', 'music', 'Germany', '1957'),
('David', 'Benioff', 'producer',  'USA', '1970'),
('David', 'Benioff', 'writer',  'USA', '1970');


insert into Role_Bio (role_first_name, role_last_name, role_function, nationality,
year_birth, year_died) values
('Peter', 'Benchley', 'writer', 'USA', '1946', '2006'),
('Gordon', 'Willis', 'cinematographer', 'USA', '1931', '2014'),
```

```
('Richard', 'Marquand', 'director', 'Wales', '1937', '1987');


insert into Film_Role (film_ID, role_first_name, role_last_name, role_function)
values
(63102, 'Christopher', 'Nolan', 'director'),
(63102, 'Christopher', 'Nolan', 'writer'),
(63103, 'Christopher', 'Nolan', 'director'),
(63103, 'Christopher', 'Nolan', 'writer'),
(63103, 'Jonathan', 'Nolan', 'writer'),
(63101, 'Frank', 'Darabont', 'director'),
(63101, 'Stephen', 'King', 'writer'),
(63112, 'Darla', 'Anderson', 'producer'),
(63112, 'Lee', 'Unkrich', 'editor'),
(63120, 'Peter', 'Benchley', 'writer'),
(63114, 'Rajkumar', 'Hirani', 'director'),
(63117, 'Hildur', 'Guðnadóttir', 'music'),
(63113, 'Makoto', 'Shinkai', 'director'),
(63113, 'Makoto', 'Shinkai', 'writer'),
(63113, 'Makoto', 'Shinkai', 'art'),
(63111, 'Laurie', 'MacDonald', 'producer'),
(63116, 'Hans', 'Zimmer', 'music'),
(63102, 'Hans', 'Zimmer', 'music'),
(63108, 'David', 'Benioff', 'producer'),
(63118, 'Richard', 'Marquand', 'director');

insert into Film_Detail (film_ID, time_start, time_spend, review, rating) values
(63120, '2022-06-07 23.15.16', 139, 'bagus sekali', 5),
(63117, '2022-06-23 19.20.21', 120, 'joker sangat seram', 4),
(63115, '2022-07-08 08.56.58', 127, 'batman keren', 5),
(63114, '2022-06-25 12.34.56', 165, 'mengharukan', 5),
(63118, '2022-06-10 20.30.40', 113, 'nostalgia film lama', 3),
(63107, '2022-06-21 23.34.11', 48, 'overrated', 4),
(63112, '2022-06-29 19.44.59', 98, 'ending yang baik', 5),
(63105, '2022-06-03 10.09.03', 168, 'agak vulgar', 4),
(63104, '2022-06-08 15.02.12', 132, 'aksi menegangkan', 5),
(63103, '2022-06-26 03.12.44', 165, 'saya ingin ke luar angkasa', 5),
(63106, '2022-06-19 07.38.26', 46, 'menonton bersama anak', 5),
(63120, '2022-06-25 12.34.56', 103, 'menegangkan', 4),
(63114, '2022-06-30 12.10.01', 161, 'memotivasi', 5);

insert into Film_Detail (film_ID, time_start, time_spend, rating) values
(63101, '2022-06-22 22.30.08', 123, 5),
(63113, '2022-07-07 12.41.22', 98, 5),
(63114, '2022-06-22 09.21.02', 160, 5),
(63104, '2022-06-30 04.15.59', 119, 3);

insert into Film_Detail (film_ID, time_start, time_spend) values
(63119, '2022-06-17 13.12.11', 160),
(63109, '2022-06-04 23.21.22', 41),
(63116, '2022-06-14 16.43.27', 146);
```

[5] Terdapat dua kesalahan pada waktu perancangan.

a. Setiap insan perfilman ternyata dapat memiliki lebih dari satu kewarganegaraan. Ubahlah model fisik sehingga dapat mengakomodasi perubahan tersebut, serta lengkapi dengan contoh data yang sesuai (5 insan perfilman dengan kewarganegaraan lebih dari satu)

b. Setiap film seharusnya memiliki informasi rumah produksi dan negara yang memproduksinya. Ubahlah model fisik sehingga dapat mengakomodasi perubahan tersebut, serta lengkapi dengan contoh data yang sesuai.

Jawaban:

```
-- 3 a
alter table Role_Bio
add second_nationality varchar(20);

update Role_Bio
set second_nationality = 'Indonesia'
where role_last_name = 'Nolan';

update Role_Bio
set second_nationality = 'Indonesia'
where role_last_name = 'Benioff';

update Role_Bio
set second_nationality = 'Indonesia'
where role_last_name = 'Shinkai';

update Role_Bio
set second_nationality = 'Indonesia'
where role_last_name = 'Darabont';

-- 3 b
alter table Film_Bio
add production_house varchar(50);

alter table Film_Bio
add country_released varchar(30);

update Film_Bio
set production_house='FamilyVideo Production';

update Film_Bio
set country_released='Indonesia';
```

4. SQL [45]

[5] Karena sistem akan menampung puluhan ribu film dan ratusan ribu insan perfilman, maka diperlukan objek basis data yang dapat mempercepat pencariannya. Tuliskan SQL untuk membuat objek basis data yang dapat mempercepat pencarian judul film dan nama insan perfilman.

Jawaban:

```
-- 4. objek basis data
create index idx_film
on Film_List(title);

create index idx_role
on Role_Bio(role_first_name, role_last_name);
```

[5] Buatlah daftar film yang berisi kode film, judul film, panjang film, rumah produksi dan negaranya, tahun rilis, yang diurutkan berdasarkan tahun rilis.

Contoh hasil:

```
ID | TITLE | DURATION | PRODUCTION_HOSE | YEAR
tt0372784 | Batman Begins | 140 | Warner Bros, US | 2005
tt6723592 | Tenet | 150 | Warner Bros, US | 2020
```

Jawaban:

```
-- 4. urut tahun rilis
select Film_List.film_ID, Film_Bio.title, Film_Bio.duration,
Film_Bio.production_house, Film_Bio.country_released, Film_Bio.year_released
from Film_List
right join Film_Bio on Film_List.title = Film_Bio.title
order by Film_Bio.year_released;
```

[10] Buatlah daftar 5 film terbaik berdasarkan nilai rata-rata rating dari anggota yang sudah menontonnya, dan dengan jumlah anggota yang memberikan rating lebih dari suatu nilai tertentu
Contoh hasil:

```
ID | TITLE | AVG(RATING) | SUM(MEMBER)
tt0372784 | Batman Begins | 4.2 | 5
tt6723592 | Tenet | 4.0 | 3
...
```

Jawaban:

```sql
-- 4. daftar 5 film terbaik
select Film_Detail.film_ID, Film_List.title, avg(rating), count(member_ID)
from Film_Detail
inner join Film_List on Film_Detail.film_ID = Film_List.film_ID
group by film_ID
order by count(member_ID) desc
limit 5;
```

[10] Buatlah daftar film yang sudah ditonton oleh pengguna tertentu yang berisi informasi kode film, judul film, rating rata-rata dari semua pengguna, yang diurutkan berdasarkan waktu menontonnya dari yang paling baru.
Contoh hasil:

```
ID | TITKE | AVG(RATING) | WATCH_TIMESTAMP
tt6723592 | Tenet | 4.0 | 2022-06-01 20:21:13
tt0372784 | Batman Begins | 4.2 | 2022-05-20 19:15:23
...
```

Jawaban:

```sql
-- 4. urut waktu menonton paling baru
select Film_Detail.film_ID, Film_List.title, avg(rating), Film_Detail.time_start
from Film_Detail
inner join Film_List on Film_Detail.film_ID = Film_List.film_ID
group by film_ID
order by Film_Detail.time_start desc
```

[10] Buatlah daftar member yang sudah dan belum memberikan rating film & review, yang diurutkan berdasarkan id keanggotaannya.

Contoh hasil:

```
ID         | NAME    | FILM    |  RATING   | REVIEW
————————————————————————————————————————————
AB1001     | Jack    | BATMAN |      |
ZA1029     | Deni    | Tenet   |   4   |
CL1901     | Joe     | Superman |  5   | Very good superhero
...
```

Jawaban:

```sql
-- 4. daftar member yang sudah dan belum rating
select Film_Detail.member_ID, Member_List.email, Film_List.title, Film_Detail.rating,
Film_Detail.review
from Film_Detail
inner join Member_List on Film_Detail.member_ID = Member_List.member_ID
inner join Film_List on Film_Detail.film_ID = Film_List.film_ID
order by Member_List.member_ID;
```

[5] Buatlah daftar film yang diurutkan berdasarkan nilai rating film dan review film.
Contoh hasil:

```
ID         | TITLE    |  RATING  | REVIEW
———————————————————————————————————————————
tt9393488 | Superman   |    5    | Very good superhero
tt9393488 | Tor        |    5    | Good film
tt6723592 | Tenet      |    4    |
tt0372784 | Saving private Ryan | 4   |
...
```

Jawaban:

```sql
-- 4. urut berdasarkan rating film
select Film_Detail.film_ID, Film_List.title, Film_Detail.rating, Film_Detail.review
from Film_List
inner join Film_Detail on Film_List.film_ID = Film_Detail.film_ID
order by Film_Detail.rating desc, Film_Detail.review desc;
```