**LAPORAN AKHIR  
PRAKTIKUM MANAJEMEN BASIS DATA**



**DISUSUN OLEH:**

|  |  |  |
| --- | --- | --- |
| **NIM** | **:** | **124240005** |
| **NAMA** | **:** | **SYAHRUL ALFIANSYAH** |
| **KELAS** | **:** | **SI-A** |
| **ASISTEN** | **:** | **KESHA AZKA AFLENI** |
|  |  | **ARJUNANTAKA SOETIA AJI SUNGKONO** |

**PROGRAM STUDI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK INDUSTRI**

**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN” YOGYAKARTA**

**2024**

# HALAMAN PENGESAHAN

**LAPORAN AKHIR PRAKTIKUM**

**MANAJEMEN BASIS DATA**



Disusun Oleh:

Telah Diperiksa dan Disetujui oleh Asisten Praktikum Manajemen Basis Data

Pada Tanggal: ……………………..

|  |  |
| --- | --- |
| *Syahrul Alfiansyah* | 124240005 |

|  |  |
| --- | --- |
| **Menyetujui, Asisten Praktikum**  **Kesha Azka Afleni**  **124220023** | **Asisten Praktikum**  **Arjunantaka Soetia Aji Sungkono**  **124220062** |

# DAFTAR ISI

[HALAMAN PENGESAHAN ii](#_Toc183529739)

[DAFTAR ISI iii](#_Toc183529740)

[TUGAS 1 ENTITY RELATIONSHIP DIAGRAM (ERD) 1](#_Toc183529741)

[1.1 Studi Kasus 1](#_Toc183529742)

[1.2 ERD 1](#_Toc183529743)

[1.3 Catatan Revisi 2](#_Toc183529744)

[1.4 Revisi ERD 2](#_Toc183529745)

[TUGAS 2 RELASI ANTAR TABEL (RAT) 3](#_Toc183529746)

[2.1 RAT 3](#_Toc183529747)

[2.2 Catatan Revisi 3](#_Toc183529748)

[2.3 Revisi RAT 3](#_Toc183529749)

[TUGAS 3 CREATE, INSERT 4](#_Toc183529750)

[3.1 Query, Screenshot, dan Penjelasan 4](#_Toc183529751)

[3.2 Catatan Revisi 9](#_Toc183529752)

[3.3 Revisi Tugas 3 9](#_Toc183529753)

[TUGAS 4 PERINTAH SQL & PERINTAH SQL LANJUTAN 10](#_Toc183529754)

[1 Query, Screenshot, dan Penjelasan 10](#_Toc183529755)

[4.2 Catatan Revisi 15](#_Toc183529756)

[4.3 Revisi Tugas 4 15](#_Toc183529757)

[TUGAS 5 AGREGAT,NUMERIK,STRING 16](#_Toc183529758)

[5.1 Query, Screenshot, dan Penjelasan 16](#_Toc183529759)

[5.2 Catatan Revisi 19](#_Toc183529760)

[5.3 Revisi Tugas 5 19](#_Toc183529761)

# TUGAS 1 ENTITY RELATIONSHIP DIAGRAM (ERD)

Membuat Entity Relationship Diagram menggunakan draw.io Dimana saya menentukan studi kasus yang saya ambil dan kemudian menentukan apa saja entitas, atribut dan relasi nya

## Studi Kasus

Dalam ERD Untuk Bioskop, Entitas Utama Yang Dipilih Adalah *Pemesan*, *Tiket*, *Studio*, Dan *Film*. *Pemesan* Mencakup Informasi Seperti Nama, Kontak, Dan ID Pelanggan, Sedangkan *Tiket* Menyimpan Data Tentang Jadwal Tayang, Posisi Kursi, Dan Harga. *Studio* Mencakup Atribut Seperti Nomor Studio, Kapasitas, Dan Tipe Studio, Sementara *Film* Mencatat Judul, Durasi, Dan Genre. Entitas Ini Mencerminkan Aktivitas Utama Bioskop.

Hubungan Antar Entitas Diatur Berdasarkan Alur Pemesanan Tiket. *Pemesan* Memiliki Hubungan Satu-Ke-Banyak Dengan *Tiket*, Karena Satu Pemesan Dapat Membeli Lebih Dari Satu Tiket. *Tiket* Terhubung Dengan *Studio* Dan *Film* Melalui Atribut Seperti Jadwal Tayang. Satu *Film* Dapat Ditayangkan Di Beberapa *Studio*, Dan Satu *Studio* Dapat Memiliki Banyak Tiket Yang Terjual. Relasi Ini Menghubungkan Data Operasional Dengan Baik.

ERD Yang Dirancang Akan Menampilkan Entitas, Atribut, Dan Relasi Dengan Struktur Yang Jelas. Sebagai Contoh, Informasi Pembelian Tiket Oleh Pemesan Dapat Ditelusuri Berdasarkan ID Tiket. Dengan Desain Yang Terorganisasi, Sistem Informasi Bioskop Mampu Mengelola Transaksi, Jadwal Film, Dan Alokasi Studio Secara Efisien, Mendukung Kelancaran Operasional Bioskop.

## ERD

1. Entitas

• Pemesan

• Tiket

• Studio

• Film

1. Atribut

• Pemesan (email, No.hp, tgl.lahir, umur, nama, nama belakang, nama depan)

• Tiket (harga, stok, id.booking, email, id\_studio, kursi, id\_kursi, No.kursi)

• Studio (judul film, kursi, id\_kursi, No. kursi, id.studio)

• Film (judul film, durasi, genre)

1. Kardinalitas

• Pemesan-Tiket many to many (M-N)

• Tiket-Studio one to many (1-N)

• Studio-Film one to many (1-N)

|  |
| --- |
|  |

## Catatan Revisi

1. Memiliki lebih dari 1 primary key
2. Jika ada ralasi nya many to many makan dijadikan table batu (entitas asosiatif)
3. Kesalahan menentukan primary key

## Revisi ERD

* saya mengubah relasi membeli menjadi entitas asosiatif dimana membeli sekarang menjadi sebuah table dengan atribut (id\_order, tgl\_order, harga, email (sebagai foreign key), no\_hp)
* menghapus atribut kunci di table yang memiliki lebih dari satu atribut kunci
* di table studio saya ubah attibut kuncinnya yang awal nya kursi menjadi ke id\_studio

# TUGAS 2 RELASI ANTAR TABEL (RAT)

Membuat relasi anter table sesuai dengan studi kasus yang di pilih

## RAT

Melanjutkan erd di atas

* Relasi antar table dimana table pemesan memiliki primary key email table ini tidak memiliki foreign key
* Kemudian ada table tiket memiliki primary key id\_booking dan memiliki foreign key email untuk menghubungkan ke table pemesan
* Kemudian ada table studio memiliki primary key kursi dan memiliki foreign key id\_booking
* Kemudian ada table film memiliki primary key judul\_film table ini tidak memiliki foreign key

|  |
| --- |
|  |

## Catatan Revisi

* + - 1. Kesalahan menentukan kardinalitas antar table

## Revisi RAT

* Saya menambahkan table membeli sehingga pemesan sakarang berelasi dengan table membeli dengan kardinalitas many to many.
* Dan table membeli berelasi dengan table tiket dengan kardinalitas one to one

# TUGAS 3 CREATE, INSERT

# 

Membuat database dan table sesuai dengan studi kasus yang sudah kita pilih di mysql dengan dan dari yang sudah di revisi.

|  |
| --- |
|  |

## Query, Screenshot, dan Penjelasan

Membuat data base

1. Database bioskop

Query :

|  |
| --- |
| create database bioskop;  (membuat database) |

Hasil Query

|  |
| --- |
|  |

Membuat table

1. Table pemesan

Query :

Tabel 3.1.1 pemesan

|  |
| --- |
| create table pemesan (  email varchar (64) NOT NULL PRIMARY KEY,  -> tgl\_lahir date,  -> no\_hp varchar (13),  -> nama varchar (50),  -> umur int (3)); |

Hasil Query

|  |
| --- |
|  |

Gambar 3.1.1 pemesan

1. Table membeli

Query :

Tabel 3.1.2 membeli

|  |
| --- |
| create table membeli (  -> id\_order varchar (12) not null primary key,  harga int (9),  tgl\_order date,  -> email varchar (64),  -> no\_hp varchar (13)); |

Hasil Query

|  |
| --- |
|  |

Gambar 3.1.2 pemesan

1. Table tiket

Query :

Tabel 3.1.3 tiket

|  |
| --- |
| create table tiket (  -> id\_booking varchar (20) not null primary key,  -> stok int (3)  -> harga int (9),  -> email varchar (64),  -> kursi varchar(3),  -> id\_studio varchar (5),  -> id\_film varchar(30)  -> judul\_film varchar(20),  -> id\_order varchar (12)); |

Hasil Query

|  |
| --- |
|  |

Gambar 3.1.3 tiket

1. Table studio

Query :

Tabel 3.1.4 tiket

|  |
| --- |
| create table studio (  -> id\_studio varchar (5) not null primary key,  -> kursi varchar (3),  -> id\_booking varchar (20),  -> judul\_film varchar (20),  -> id\_film varchar (30)); |

Hasil Query

|  |
| --- |
|  |

Gambar 3.1.4 tiket

1. Table film

Query :

Tabel 3.1.5 film

|  |
| --- |
| create table film (  -> id\_film varchar (30) not null primary key,  -> durasi time,  -> genre varchar (10), -> judul\_film varchar (20),  -> id\_studio varchar (5));  (membuat table) |

Hasil Query

|  |
| --- |
|  |

1. Gambar 3.1.5 film

Menghubungkan foreign key menggunkan constraint

1. Query :

|  |
| --- |
| ALTER TABLE membeli ADD CONSTRAINT fkemail FOREIGN KEY (email) REFERENCES pemesan (email);  (Menghubungkan foreign key menggunkan constraint) |

Hasil Query

|  |
| --- |
|  |

Insert table atau mengisi data di dalam table

1. Query :

|  |
| --- |
| INSERT INTO pemesan VALUES ('syahrul@gmail.com', '2006-03-21', '082323433434', 'syahrul alfiansyah', '18'),('asep@gmail.com', '2006-12-23', '083232453727', 'asep muslim', '29'),('lia@gmail.com', '2007-11-22', '081226561811', 'lia maya', '18'),  ('adul@gmail.com' '2001-06-2','081299123312',' 'adul cecep', '22'),('malik@gmail.com', '2004-09-12', '081124789819', 'malik akbar', '22');  (Insert table atau mengisi data di dalam table) |

Hasil Query

|  |
| --- |
|  |

## Catatan Revisi

* Tidak mengetik query

## Revisi Tugas 3

# TUGAS 4 PERINTAH SQL & PERINTAH SQL LANJUTAN

Membuat informasi yang berguna menggunakan perintah SQL dan perintah lanjutan dari database yang sudah saya buat

## Query, Screenshot, dan Penjelasan

* + - 1. **Operator Kisaran**

Query :

|  |
| --- |
| Select judul\_film, stok, id\_studio from tiket where stok between 1 and 50; |

|  |
| --- |
|  |
| Menampilkan judul film dan id\_studio dengan syarat stok terdapat 1 sampai dengan 50 yang tersedia. |

Query :

|  |
| --- |
| Select judul\_film, stok, id\_studio from tiket where stok not between 1 and 50; |

|  |
| --- |
|  |
| Menampilkan judul film dan id\_studio dengan syarat stok diluar atau melebihi 1 sampai dengan 50 yang tersedia. |

* + - 1. **Operator List (OPERATOR IN)**

Query :

|  |
| --- |
| Select judul\_film, genre from film where genre in (‘romance’); |

|  |
| --- |
|  |
| Menampilkan judul film dengan bergenre “romance” |

* + - 1. **Operator String (LIKE)**

Query :

|  |
| --- |
| Select nama, email from pemesan where nama like ‘a%’; |

|  |
| --- |
|  |
| Menampilkan nama dan email dari pemesan tiket bioskop dengan nama yang di awali dengan huruf ‘a’ |

Query :

|  |
| --- |
| Select nama, email from pemesan where nama like ‘%p’; |

|  |
| --- |
|  |
| Menampilkan nama dan email dari pemesan tiket bioskop dengan nama yang di akhiri dengan huruf ‘p’ |

Query :

|  |
| --- |
| Select nama, email from pemesan where nama like ‘%li%’; |

|  |
| --- |
|  |
| Menampilkan nama dan email dari pemesan tiket bioskop dengan nama mengandung karakter ‘li’ |

* + - 1. **Klausa Order By**

Query :

|  |
| --- |
| Select \* from film order by id\_studio desc; |

|  |
| --- |
|  |
| Menampilkan seluruh isi table film yang di urutkan berdasarkan kolom ide\_studio  yang terbesar |

Query :

|  |
| --- |
| Select \* from film order by id\_studio asc; |

|  |
| --- |
|  |
| Menampilkan seluruh isi table film yang di urutkan berdasarkan kolom ide\_studio yang terkeci |

* + - 1. **Perintah SQL lanjutan**

1. Cross Join

Query :

|  |
| --- |
| Select \* from studio cross join tiket; |

|  |
| --- |
|  |
| Penggabungan tabel tanpa sebuah syarat = menampilkan semua isi table |

1. Natural Join/Inner Join

Query :

|  |
| --- |
| Select s.kursi, t.email, t.judul\_film, from studio inner join tiket t on s.id\_booking = t.id\_booking; |

|  |
| --- |
|  |
| Menampilkan data dari tabel *studio* dan *tiket* di mana terdapat kecocokan pada kolom id\_booking. Hanya data yang memiliki pasangan di kedua tabel yang akan ditampilkan |

1. Left Outer Join

Query :

|  |
| --- |
| Select s.kursi, t.email, t.judul\_film, t.stok, t.harga, t.id\_film from studio s left outer join tiket t on s.id\_booking = t.id\_booking; |

|  |
| --- |
|  |
| Menampilkan semua data dari tabel *studio*, meskipun tidak ada kecocokan di tabel *tiket*. Jika tidak ada pasangan, nilai kolom dari tabel *tiket* akan bernilai *NULL*. |

1. Right Outer Join

Query :

|  |
| --- |
| Select s.kursi, t.email, t.judul\_film, t.stok, t.harga, t.id\_film from studio s right outer join tiket t on s.id\_booking = t.id\_booking; |

|  |
| --- |
|  |
| Menampilkan semua data dari tabel *tiket*, meskipun tidak ada kecocokan di tabel *studio*. Jika tidak ada pasangan, nilai kolom dari tabel *studio* akan bernilai *NULL*. |

1. Full Outer Join

Query :

|  |
| --- |
| Select s.kursi, t.email, t.judul\_film, t.stok, t.harga, t.id\_film from studio s left outer join tiket t on s.id\_booking = t.id\_booking union Select s.kursi, t.email, t.judul\_film, t.stok, t.harga, t.id\_film from studio s right outer join tiket t on s.id\_booking = t.id\_booking; |

|  |
| --- |
|  |
| Menggabungkan hasil dari *Left Outer Join* dan *Right Outer Join*. Dengan menggunakan perintah UNION, query ini memastikan semua data dari kedua tabel (*studio* dan *tiket*) ditampilkan, baik yang memiliki pasangan maupun yang tidak. Jika data tidak memiliki pasangan di tabel lain, kolom yang tidak terhubung akan bernilai *NULL* |

## 4.2 Catatan Revisi

* Pada perintah SQL lanjutan tidak menulis penjelasan

## 4.3 Revisi Tugas 4

# TUGAS 5 AGREGAT,NUMERIK,STRING

Membuat informasi yang berguna menggunakan perintah SQL dan perintah lanjutan dari database yang sudah saya buat

## Query, Screenshot, dan Penjelasan

1. Fungsi Agregat

* Count

|  |
| --- |
| Select count(\*) from pemesan where tgl\_lahir like ‘2006%’; |

|  |
| --- |
|  |
| Query ini bertujuan untuk menampilkan berapa banyak data pemesan yang lahir pada tahun 2006 |

* Sum

|  |
| --- |
| select sum(stok) from tiket where harga = 49999; |

|  |
| --- |
|  |
| Query ini menunjukkan jumlah semua tiket upin ipin yang masih tersedia |

* Average

|  |
| --- |
| select avg(stok) as 'rata-rata tiket yang tersedia pada bulan ini' from tiket; |

|  |
| --- |
|  |
| Qurey ini menampilkan rata-rata tiket yang tersedia pada jangka waktu 1 bulan |

* Max

|  |
| --- |
| select max(stok) from tiket where harga = '49999'; |

|  |
| --- |
|  |
| Query ini menampil stok tiket terbanyak di mana harga tiket nya 49999 |

* Min

|  |
| --- |
| select min(tgl\_lahir) as 'abang abangan' from pemesan; |

|  |
| --- |
|  |
| Query ini menampilkan pemesan tiket tertua |

1. Fungsi Numerik

* Mod

|  |
| --- |
| select mod(stok, 5) from tiket; |

|  |
| --- |
|  |
| Menampilkan sisa bagi jika semua stok tiket di bagi 5 |

* Trunc

|  |
| --- |
| select truncate (avg(stok),2) from tiket; |

|  |
| --- |
|  |
| Memotong nilai yang sebelum nya 46.8000 menjadi 46.80 karena truncate berguna untuk memotong nilai di belakang koma sesuai dengan yang kita mau |

1. Implementasi select tanggal

* Menampilkan tanggal pembelian dan total harganya

|  |
| --- |
| SELECT DATE\_FORMAT(tgl\_order, '%d-%m') AS bulan,sum(harga) as 'total harga' from membeli order by bulan; |

|  |
| --- |
|  |
| Tetapi kenapa total harga muncul 49999 saja karena pada bulan tidak ada yang melalukan pembelian yang sama atau pemesan membeli tiket pada bulan-bulan yang berbeda |

* Menampilkan nama sesuai dengan umur yang di tentukan

|  |
| --- |
| SELECT nama, DATE\_FORMAT(tgl\_lahir, '%d-%m-%Y') AS 'tanggal lahir' FROM pemesan where umur = 18; |

|  |
| --- |
|  |
| Menampilkan nama pemesan sesuai dengan umur yang di tentukan |

## Catatan Revisi

* Kurang jelas pada perintah MIN
* Kurang data pada perintah MOD

## Revisi Tugas 5

* + - 1. Mod

|  |
| --- |
| select mod(stok, 5) from tiket; |

|  |
| --- |
|  |
| Menampilkan sisa bagi jika semua stok tiket di bagi 5 |

* + - 1. Min

|  |
| --- |
| SELECT \* FROM pemesan WHERE umur = (SELECT MIN(umur) FROM pemesan); |

|  |
| --- |
|  |
| Query ini menampilkan pemesan tiket termuda |